Université de Montréal

DÉVELOPPEMENT D'UNE INTERFACE GRAPHIQUE D'AIDE AUX ACTEURS DE LA CONCEPTION ARCHITECTURALE

Par

Djebbar Benmoumene

Faculté de l'aménagement

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maîtrise ès sciences appliquées
en aménagement
Option : conception, modélisation et fabrication assistée par ordinateur

septembre 2005

Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé :

DÉVELOPPEMENT D'UNE INTERFACE GRAPHIQUE D'AIDE AUX ACTEURS DE LA CONCEPTION ARCHITECTURALE

présenté par :

Djebbar Benmoumene

a été évaluée par un jury composé des personnes suivantes :

T. Ticafi .....Président rapporteur

Giovanni De Paoli.....Directeur de recherche

T. Dosta .....Membre du jury

Résumé

L'objectif de cette recherche est de proposer un outil d'aide aux acteurs de la conception architecturale. Il a pour fonction d'assister tout acteur dans le cadre des compétences à générer un modèle d'objet architectural.

La conception architecturale devient de plus en plus complexe par les nouvelles procédures de mises en œuvre et de réalisation. C'est un domaine qui fait appel à de multiples spécialistes, à de nouvelles compétences et de nouveaux métiers.

Aujourd'hui, comme dans tous les domaines, l'outil informatique est inévitablement, l'outil de base de ces acteurs de la conception architecturale. Il est le moyen de travail et de communication entre tous les professionnels.

Si pour la plus part des spécialités, l'ordinateur est exploité de façon optimale, son utilisation dans la conception architecturale est loin des attentes des acteurs de ce domaine. Le constat enregistré sur l'écart existant entre son exploitation et les possibilités informatiques nous a entraîné à poser la question de recherche :

Comment l'outil informatique peut-il servir un acteur de la conception architecturale, dans le cadre de ses compétences ?

Nous avons alors fixé comme hypothèses qu'il possible à tout acteur de la conception architecturale d'utiliser l'ordinateur dans le cadre de ses compétences si nous instrumentons les procédures de création des modèles objets architecturaux.

Pour entreprendre notre recherche, nous avons commencé par la collecte d'information sur certaines notions liées à notre sujet de recherche. Ces notions nous, ont permis de dégager un certain nombre de conclusion que nous avons exploité comme socle de notre recherche.

Ainsi, nous avons débuté par comprendre le sens général de la conception et celui de la conception architecturale où nous avons passé en revue quelques tendances liées à l'activité de la conception architecturale. Nous avons retenu l'existence d'au moins deux visions distinctes : La visions de P.Boudon et celle d'un ensemble d'autres chercheurs (Prost, Collan, Tidafy… etc). Ces deux visions s'opposent si la conception est du domaine exclusif de l'architecte ou c'est une activité collective de tous les acteurs participants à la conception.

Comme deuxième niveau, nous avons entamé une étude critique sur les outils de figurations utilisés dans la conception architecturale (les outils traditionnels, le DAO et la CAO). Ce travail nous a permis de mettre en évidence les avantages et les limites de chacun de ces outils par rapport aux attentes des acteurs de la conception architecturale.

L'étude des deux chapitres précédents nous a permis de retenir deux conditions qu'un outil devrait satisfaire pour servir un acteur de la conception architecturale, il s'agit :

- Etre en conformité avec son profil.
- Répondre à ses intentions et ses besoins.

Ces deux conclusions retenues nous ont entraîné à proposer une méthode qui se base sur l'instrumentation des procédures de création des modèles d'objets architecturaux. Pour y faire, nous avons réuni deux approches :

- l'approche procédurale qui consiste à décrire les objets architecturaux par leurs processus de leurs créations.

- L'approche visuelle qui consiste à créer des fenêtres de dialogue qui répondent à la compétence informatique de chacun de ses acteurs.

Afin de vérifier nos hypothèses, nous avons opté pour l'escalier comme étude de cas. Ce choix est motivé par la possibilité qu'offre l'approche procédurale de décrire la variété de types d'escaliers existants, tout en évitant la redondance de réécriture de l'information.

Mots-clés : conception architecturale, modèle, CAO, DAO, acteur de la conception architecturale, approche procédurale, approche visuelle.

Abstract

The aim of this research is to propose a computational assistant that would be an efficient tool to generate architectural models. This tool by its flexible definition will enable all design actors and partners in their related specialized fields.

Our study takes for a point of departure the ideal that architectural design getting increasingly complex due to the new emerging tools and the continuously developed means. The design activity is now involving so many actors from a very wide range of specialization.

The computer aided-design in the field of architecture is however far away from being satisfactory to all designers that interfere in the design process. This remark had led us to formulate the following question:

How could computer tools be used to serve better the design actors to accommodate their different interests and domains during the design development processes?

Our study is then based on the hypothesis that every design actor can make use of the computer if we can success in adapting the creating procedures that result in architectural models to their different intents and profiles.

We undertook our research by developing certain aspects related to our topic. Thus we have developed the concepts of "design "and architectural design ", where we noticed that there are at least two major visions: The one led by (Boudon) who stands that the architectural design is the exclusive domain of architect. The other is led by many authors (such as Prost, Collan and Tidafy) who define the design as a collective activity that reunify all intervening actors.

At the other level, we studied the common tools (currently used in architectural design). Our intent in this part was to fix all advantages and limits of the design practice and its accordance to all actors.

Our study brought us to think that every design tool to be efficient and useful should satisfy two conditions.

- The tool must be as much s possible the design actor's profiles.
- Accommodate the intents and the needs of the actors.

Based on the two proper observations, we deduced from our study, we have built our own methodology on the main idea of adapting the creating procedures of architectural models that consist on:

- Procedural's approach that describe architecture's models throughout their creation processes.
- Visual approach on the basis of interacting dialog boxes that take into account and operate the design data.

To verify our hypotheses and validate thoughts, we choose stairs like a case of study. This choice is motivate by the possibility that given to us procedural approach in describing the variety of stairs without information that's redundancy.

Through our method, we had verified, that actors who take part in design activity can use the computer according to their goals and intents.

Keywords: Design, Procedural approach, visual approach, model, design actor.

Table des matières

I- CONCEPTUALISATION DE LA RECHÉRCHE

Liste des tableaux

Liste des figures

A ma mère

Remerciements

Merci à M. Giovanni De Paoli d'avoir accepter de diriger ce mémoire et de m'avoir permis de bénéficier de ses connaissances et de ses expériences. Je lui exprime ma gratitude pour sa disponibilité à répondre à mes nombreuses questions lors des visites prévues ou imprévues dans son bureau.

Je remercie également tous les enseignants que j'ai connus lors de cette maîtrise.

Finalement à tous ceux qui ont contribué de prés ou de loin à l'élaboration de ce mémoire.

I- CONCEPTUALISATION DE LA RECHÉRCHE

I.1. PRÉSENTATION DU PROBLEME DE RECHERCHE :

I.1.1. INTRODUCTION :

La substitution de l'outil informatique à la feuille de papier a permis aux usagers de l'ordinateur de disposer d'un outil de dialogue permanent dans leurs pratiques quotidiennes. La rapidité enregistrée dans l'évolution des technologies de l'information et de la communication durant le siècle dernier a eu son impact sur notre vie. En un temps très restreint, l'ordinateur a connu une propagation considérable. Il a graduellement perdu son rang de machine exceptionnelle pour devenir l'outil technique de tous les domaines. Toute personne est aujourd'hui, concernée par l'usage, désormais banalisé, des outils informatiques.

Toutes les sciences ont prouvé que l'informatique est une occasion pouvant ouvrir des fenêtres sur de nouveaux horizons. Seulement, malgré les potentialités qu'offre l'outil informatique, nous remarquons, encore, que la tâche assignée à cet outil d'aider les utilisateurs dans leurs activités quotidiennes est loin d'être à son optimum. Si l'intégration des nouvelles technologies a donné pleine satisfaction pour la plupart de certains domaines, beaucoup d'autres sont loin de les exploiter dans leurs véritables dimensions. On persiste, parfois encore, à adapter les mêmes méthodes héritées des outils traditionnels.

Le progrès technologique a mis en évidence de nouvelles dimensions que le support traditionnel ne pouvait pas permettre. Le développement de l'informatique a donné naissance à d'autres modes de relations, il a engendré de nouveaux modes de questionnement et de nouvelles formes de lectures et d'utilisation, donc de nouvelles manières de penser. Aujourd'hui, la représentation de l'information change de support et s'affiche sur un écran. Un changement de support qui implique de nouvelles formes de lecture et de raisonnement.

Ainsi, dans le domaine de la conception architecturale, l'adoption de l'ordinateur demeure loin des attentes de ses acteurs. Parfois, on persiste à utiliser les mêmes pratiques héritées des méthodes classiques et parfois, encore, on utilise des modes de descriptions difficilement compréhensibles ou accessibles par les divers acteurs de la conception architecturale.

Le développement de l'informatique ressuscite actuellement le problème de la représentation du bâtiment sous une forme permettant une exploitation efficace de l'information (C.Parisel, 1990). Malgré toutes les possibilités qu'offre l'outil informatique, l'ordinateur est limité à de simples tâches de créer, visualiser et produire des dessins. Cette exploitation limitée évite aux acteurs de la conception architecturale de disposer de véritables outils d'assistance dans leurs pratiques quotidiennes.

Ce constat nous a motivé a savoir comment l'outil informatique peut-il servir de support d'aide aux divers acteurs de la conception architecturale?

I.1.2. PROBLEMATIQUE :

La conception architecturale fait de plus en plus appel à des connaissances et à des compétences diversifiées. Elle devient un domaine de plus en plus complexe par les procédures de mises en oeuvre, par la multiplication des spécialistes, par l'appel à de nouvelles compétences et de nouveaux métiers. Elle exige que ces savoirs et ces compétences soient coordonnés le mieux possible.

Habituellement les acteurs se communiquent sur la conception via des outils de description traditionnels (texte, dessin...etc.). Aujourd'hui, l'ordinateur a révolutionné notre quotidien pour devenir l'outil technique de tous les domaines. Si son adoption par la plupart des spécialités a donné ses résultats, nous constatons encore que l'interaction entre l'homme et son ordinateur dans le domaine de la conception architecturale est loin d'être à son optimum. Il existe encore un grand écart entre l'exploitation actuelle de l'ordinateur et les possibilités informatiques.

L'outil informatique ne contribue pas à l'assistance de tous les acteurs de la conception. Il est encore utilisé de la façon traditionnelle en se limitant seulement à l'activité de produire, de visualiser et d'apporter certaines modifications lorsque la conception d'un projet est suffisamment avancée. Or, les possibilités de l'informatique dépassent largement cette pratique habituelle des agences architecturales. Son utilisation raisonnée et son exploitation efficace comme un véritable outil d'aide à la conception architecturale nécessite la prise en compte des possibilités informatiques actuelles par rapport aux compétences et aux besoins de chacun de ces acteurs.

C'est pourquoi, nous posons comme question de départ :

Comment l'outil informatique peut-il servir un acteur de la conception architecturale, dans le cadre de ses compétences ?

I.1.3. HYPOTHÈSE GÉNÉRALE :

Nous formulons notre hypothèse de la façon suivante:

Si nous instrumentons les procédures de création des modèles d'objets architecturaux, tout acteur de la conception architecturale peut rendre visible ces modèles d'objets sur lesquels qu'il peut communiquer.

On comprend par les modèles d'objets architecturaux, les objets architecturaux élémentaires ou «éléments d'architecture», constituant l'univers instrumental de l'architecte ((P.Quintrand, 1985).

Pour cela, notre hypothèse se base sur la possibilité d'associer deux méthodes complémentaires ou chacune apporterait un ensemble d'éléments de réponse à notre problématique:

- **Une approche pour figurer les objets : approche procédurale**

La notion de procédure est de plus en plus utilisée dans le domaine de la CAO. L'approche procédurale anticipe certaines décisions grâce à la possibilité d'enregistrer de façon intelligente non pas l'objet mais le processus de sa création. Grâce à cette approche et en se basant sur les langages de programmation, il est possible de consigner les procédures permettant de créer un modèle d'objet architectural.

- **Une approche pour faciliter le dialogue : approche visuelle :**

Dans le but d'assurer un dialogue naturel, nous devons instrumenter les procédures qui permettront à tout acteur d'une solution architecturale de générer un modèle d'objet sans connaissance du formalisme du langage de programmation utilisé.

Aujourd'hui, les interfaces graphiques sont de plus en plus utilisées pour faciliter aux usagers l'interaction avec l'ordinateur. Grâce à la programmation visuelle, il est possible d'instrumenter les procédures de création de ces modèles et permettre à tout usager de choisir et de générer une solution envisageable sans pour autant connaître la programmation.

I.1.4. OBJECTIFS DE LA PRÉSENTE RECHERCHE :

La présente recherche a pour objectif de proposer une méthode alternative d'aide à la conception. Elle vise surtout à disposer tout acteur participant à une solution architecturale d'un outil capable de l'assister à faire une copie de l'image mentale d'un objet ou d'une solution envisageable à concevoir ou à communiquer à ses partenaires sans pour autant qu'il ait une grande connaissance informatique.

Le recours à la programmation informatique est de plus en plus utilisé en architecture. Selon (P.Quintrand, 1985), si nous possédons les outils nécessaires pour manipuler l'information, une bonne programmation devrait assurer de bien définir les problèmes et la solution devrait normalement en découler. Mais, selon T.Tidafi (1996) en évoquant les limites liées à la programmation, l'interface permettant le dialogue entre l'homme et la machine peut constituer un inconvénient si nous ne choisissons pas la façon d'instrumenter ce dialogue (T.Tidafi 1996).

C'est pourquoi nous nous fixons dés le départ comme objectif : de trouver une méthode pouvant faciliter aux acteurs de la conception de rendre visible un modèle d'objet architectural sans grande connaissance informatique.

I.1.5. MÉTHODE DE TRAVAIL :

La conception architecturale fait appel à nombreux acteurs. Ils sont souvent de formation variable, de compétence différente et ont chacun leur point de vue sur l'objet architectural à concevoir.

À travers l'histoire, nous constatons que malgré l'apparition de nouveaux formats de collaboration et de communication entre les acteurs de conception, cette évolution n'a pas influencé sur la nature des échanges et des communications relatives à la conception d'un projet architectural. Aujourd'hui, l'ordinateur est l'outil de travail de tous les acteurs de la conception architecturale. Son utilisation comme moyen de communication et collaboration exige la prise en compte des attentes de ces acteurs.

Pour répondre à notre problématique et vérifier ainsi nos hypothèses, nous commencerons en premier lieu par comprendre le sens de la conception architecturale. Dans ce chapitre, nous étudierons quelques visions sur la participation des acteurs dans l'activité de la conception architecturale. Nous devons aussi faire un survol des outils utilisés dans ce domaine. Ce chapitre nous aidera à comprendre et vérifier l'utilisation de ces formats d'échanges par rapport aux profils (compétences) et des besoins des acteurs de la conception architecturale.

Dans la deuxième étape, nous devons décrire une méthode à adopter pour soumettre nos hypothèses et notre problématique à l'épreuve des faits. Ainsi, nous chercherons les moyens existants pour répondre aux conclusions qui seront retenues dans le chapitre précèdent. L'étude des diverses stratégies existantes nous permettra d'identifier les avantages et les inconvénients de chacune d'elles. Cette étude nous aidera à présenter notre stratégie et à définir les caractéristiques à retenir pour notre approche informatique.

Comme troisième étape, nous devons vérifier la cohérence de notre de la base théorique de travail. Pour cela, nous choisirons le cas de l'escalier pour plusieurs raisons :

Comme pour tout élément de construction architectural, la conception et la réalisation d'un escalier fait appel à de multiples acteurs. La nécessite de collaboration fait souvent appel à un moyen d'échanges entre ces acteurs. L'infinité de modèles d'escaliers existants rend la tache impossible aux formats traditionnels d'anticiper la description de tous ces modèles. Même les sources documentaires d'aide à la conception se limitent dans le cas d'un escalier à décrire un

seul type de modèle à travers le quel on explique comment on peut construire d'autres modèles partageant les même caractéristiques.

Aujourd'hui avec l'intégration de l'outil informatique dans le domaine de l'architecture de nouvelles formes de description et de raisonnement ont vu le jour où il est possible de permettre à un acteur de la conception architecturale dans le cadre de ses compétences de naviguer d'un modèle " escalier " à un autre sans être obliger de réécrire l'escalier.

Enfin comme quatrième étape, nous devons tirer les conclusions sur les résultats de notre expérimentation et identifier les perspectives de recherches potentielles pour d'autres applications analogues.

I.2. LA CONCEPTION ARCHITECTURALE ET SES ACTEURS :

I.2.1. INTRODUCTION :

L'avènement des outils de modélisation, de simulation et d'animation 3D comme support à la conception architecturale amorcent des changements dans l'exercice de la pratique architecturale. La relation entre la conception architecture et l'ordinateur est devenue une question d'actualité souvent posée.

Pour mieux cerner les éléments entourant notre recherche et saisir le sens de notre réflexion, nous avons convenu dans un premier lieu de définir les thématiques, notions et concepts liés à notre sujet.

Rendre compte des conséquences du recours aux outils informatiques suppose au préalable la connaissance théorique de ce que veut dire la conception architecturale. On ne peut parler d'outils d'aide à la conception sans comprendre le sens même de la conception en général et celle de l'architecture en particulier.

I.2.2. LA CONCEPTION :

Le mot «conception » désigne selon le dictionnaire : Le Robert (2001) : *« La formation d'un concept, d'une idée générale dans l'esprit»* mais aussi : *« l'action de concevoir, l'acte de l'intelligence de la pensée, s'appliquant à un objet».*

Cette définition nous montre bien qu'il est difficile de donner un seul sens au mot «conception ». Il peut prendre des significations différentes selon le contexte de son utilisation. Mais si en général, la conception est définie comme un ensemble d'actions engendrant un résultat, certains auteurs contemporains dans plusieurs domaines, ayant traité cette question, l'attribuent à toute activité intellectuelle que l'on pratique sur un objet réel ou virtuel.

Parmi ces chercheurs, Prost (1994) définit la conception comme :« *agir en pensant et penser en agissant».* A travers cette définition, il évoque ce double lien entre la connaissance et l'action. *«L'action ne va plus de soi et réclame de la réflexion, mais la pensée à elle seule ne peut transformer le monde et il faut bien qu'elle cherche les conditions de sa transcription, voir de sa métamorphose, dans l'agir».*

Dans le même ordre d'idée, pour J.L LeMoigne (1990) : *« La conception est une action cognitive finalisée. C'est la quête de solutions possibles à des problèmes artificiellement posés qui guide en permanence la démarche du concepteur : elle est par construction tâtonnante, s'auto-jalonnant d'objectifs intermédiaires, mettant en oeuvre de multiples heuristique».*

De même, H.Simon (1974) révèle l'importance du savoir et de la connaissance pour tout concepteur. *«Quiconque imagine quelques dispositions visant à changer une situation existante en une situation préférée, est concepteur. L'activité intellectuelle par laquelle sont produits les artefacts matériels n'est pas fondamentalement différente de celle par laquelle on prescrit un remède à un malade ou par la quelle on imagine un nouveau plan de vente pour une société, voir même une politique sociale pour un état. La conception, ainsi conçue, est au cœur de toute formation professionnelle. C'est elle qui fait la différence entre sciences et professionnels. Les écoles d'ingénieurs, comme les écoles d'architecture, de droit, de gestion, de médecine, les écoles normales d'enseignement, toutes sont concernées, au premier chef par le processus de la conception».*

Toutes ces définitions convergent à définir la conception comme une phase décisive ou l'avenir d'un objet est mis en étude. C'est une projection de la réalité d'un objet. Tous les chercheurs traitent la conception comme une activité intellectuelle. Ils évoquent tous l'importance de la réflexion et de la connaissance pour bien mener l'activité de la conception.

I.2.3. CONCEPTION ARCHITECTURALE ET SES ACTEURS :

I.2.3.1. INTRODUCTION :

Parmi les réflexions ayant émergées sur l'architecture durant le vingtième siècle, Un nombre important des discours gravite autour des problèmes relatifs aux pratiques, aux acteurs et aux processus de conception architecturale.

Le survol des différents écrits ayant traités la conception architecturale, nous révèle l'existence d'au moins deux visions distinctes par rapport la participation des acteurs dans l'activité de la conception architecturale :

I.2.3.2. LA CONCEPTION ARCHITECTURALE ET L'ARCHITECTUROLOGIE (VISION DE BOUDON) :

Pour P.Boudon (1992), la conception architecturale est définie par les opérations auxquelles l'architecte se livre durant son travail de conception. Afin de dissocier le travail de l'architecte des autres acteurs, il pose sur la base du modèle architecturologique, le travail de conception architecturale comme un objet de recherche ou scientifique.

Pour cela, il met en évidence les opérations qu'utilise l'architecte lorsqu'il conçoit. Selon lui, l'architecturologie se base sur 3 concepts : l'embrayage, le modèle et l'échelle.

- *L'embrayage* est l'acte qui permet de mesurer des formes architecturales.
- *Le modèle* est le volume que l'architecte crée en lui attribuant les dimensions.
- *L'échelle* est le principal concept de l'architecturologie. Elle assure les transformations afin de donner forme et dimensions aux projets en conception.

Selon P.Boudon (1992), l'architecturologie adhère à quatre points qui définissent la conception architecturale:

- Elle s'attache à des projets non conçus.
- Elle étudie les opérations mais non pas les états (c'est-à-dire l'étude d'une esquisse à une autre et non pas celle déjà faite par un architecte.
- Elle explore les opérations sous l'angle matériel et cognitif.
- Elle attache un esprit *«poïétique»* aux opérations.

Tout en considérant l'architecte comme un acteur parmi tant d'autres intervenants qui sont producteurs du projet architectural, P.Boudon estime que les autres acteurs (en dehors de l'architecte) prennent des décisions mais ne se livrent pas à la conception. En d'autres termes, nous devons faire la différence entre *«la décision»* concernant un objet conçu et *«la conception»* elle même d'un objet.

I.2.3.3. LA CONCEPTION ARCHITECTURALE COMME TRAVAIL COLLECTIF DE SES ACTEURS :

> *Les recherches récentes convergent pour définir la conception architecturale comme un processus collectif au cours duquel les points de vue sont négociés, grâce à la mise en œuvre de différents supports : dessins, plans, maquettes...*
>
> M.Callon, 1996

Contrairement à Boudon, beaucoup d'autres auteurs ayant étudiés cette question, ne limitent pas la conception architecturale uniquement à l'intervention de l'architecte mais la considèrent comme une tâche collective. La majorité des autres chercheurs (Prost, Collan, Tidafy...etc) évoque le lien étroit existant entre la conception et ses acteurs. Ces derniers sont unanimes sur le point de vue d'une création collective. Tout projet tient aux notions de négociations entre ces acteurs.

Si en général, un acteur désigne un artiste interprétant un rôle, il peut aussi signifier toute personne, tout groupe d'individus ou même une institution auxquels un rôle est assigné (C. Verrier 1999).

Dans le domaine de la conception architecturale, un acteur est toute personne ou toute administration qui prend part à la réalisation d'un projet architecturale. Le nombre d'acteurs peut varier selon la complexité de l'ouvrage à réaliser. Très souvent, le profil des acteurs est variable selon la tâche à accomplir.

R.Prost (1992) la définit comme un processus complexe où des acteurs avec des intérêts différents sont associés à un seul but qui est la solution architecturale.

Pour M.Callon (1996), il est difficile, voire impossible d'attribuer à une personne ou à un petit groupe la paternité d'un projet de conception car la conception architecturale est un résultat d'un travail collectif et d'une réalité composée, obtenue après une suite de d'échanges de vues entre les partenaires du projet en question.

Dans le même ordre d'idées, M.Callon, (1996) considère qu'il y a conception lorsqu'il n'y a pas de créateur mais un groupe qui se discute par des schémas ou des maquettes interposées. Elle s'inscrit dans un processus collectif. Pour lui, la conception n'a rien d'immatérielle, elle appartient au monde des graphiques et des stratégies de visualisation où l'ordinateur occupe une place importante.

De même, V. Le Goaziou [1] ; note que la conception est un processus inachevé qui se trouve entre deux discours :

---

[1] ; tiré de : R.Prost, (1994): Concevoir, inventer, créer : les réflexions sur les pratiques. Paris. Édition L'Harmattan.

- Le premier discours est celui qui place la conception dans une vision théorique et philosophique comme un système de procédures et d'actes rationnels, ordonnés, logiques et maîtrisés. Ce type de discours est très critique par rapport à la science, la technique et l'apport de l'industrie.

- La deuxième vision de la conception est celle qui existe chez ses acteurs (designers, fabricants, chercheurs…) et qui la décrivent comme un processus relevant du «faire humain».

Parmi ces deux discours, l'auteur préconise le deuxième. Ce dernier est plus visible si on travaille avec les acteurs de la conception en ayant la possibilité de suivre certaines de leurs réalisations. L'objet final est généralement loin des premières idées existantes au début d'un projet d'architecture.

Selon M.Callon (1996), il existe deux raisons pour lesquelles, il est vain et stérile de vouloir séparer les opérations de conception et celles de l'exécution.

- Le projet ne se réalise jamais tel que dessiné sur le plans, il dérive en subissant des changements. Très souvent, durant la réalisation, pour une raison ou une autre, les choix sont remis en cause, ce qui conduit à revenir souvent sur le projet initial et le transformer.

- La deuxième raison concerne cette dualité existante entre idées et pratiques matérielles. Même, si on étale le processus de conception dans le temps et dans l'espace pour assurer les allers et retours, les modifications et les différents changements pouvant toucher un projet, on ne pourrait pas affirmer que la phase d'exécution commencera une fois l'accord est obtenu car les transformations ne s'arrêteront jamais durant la réalisation.

M.Callon(1996) s'interroge, s'il est important que le processus de conception soit redistribué à tous les partenaires, comment assurer la coordination entre les acteurs appartenant souvent à des mondes différents? Et comment la mise en relation des points de vue s'opère-t-elle?

Selon lui, il n'existe qu'une réponse à cette interrogation qui consiste à rendre visibles et manipulables ce que le simple langage ne pourrait décrire à savoir : les espaces, les formes et les volumes. Ainsi, sans une médiation visuelle produite en permanence entre les différents partenaires, la communication des points de vue serait impossible.

Prost (1992) constate que le principe du processus de conception est fondé sur les actions des acteurs. Il identifie trois (3) registres qui constituent des sous processus d'un processus de conception : (1) les processus de formulation des problèmes, (2) les processus de formulation de solution et (3) les processus de concrétisation d'une solution conçue.

- Le premier registre tel qu'évoqué réside à cerner un problème en collectant les informations relatives à sa compréhension et son interprétation par les initiateurs du projet.

- Le deuxième consiste à rendre visible la solution en passant des mots à des solutions visibles.

- Le troisième tel que son nom l'indique, sert à concrétiser la solution.

En assimilant ces trois registres d'un processus de conception architecturale, T.Tidafi estime que les acteurs d'une solution architecturale peuvent intervenir durant plusieurs phases d'une concrétisation de cette solution (1996). Il identifie cinq (5) types de communication capables de se réaliser entres les différents acteurs d'une solution architecturale :

- Communication préparatoire : Il s'agit de toute communication qui se déroule lors du début d'un processus de conception.

- Communication pour la recherche d'une idée de solution: Lors de cette phase, les acteurs tentent de formuler une solution.

- Communication relative à la pertinence de la solution : Les acteurs examinent le bien-fondé de la solution formulée.

- Communication relative à la technique de la solution : Cette phase consiste à définir comment exécuter la solution en détaillant son aspect technique grâce à l'expérience des acteurs.

- Communication sur la concrétisation de la solution : Il s'agit de l'exécution de la solution.

Tout comme M.Callon, T.Tidafi évoque la nécessité de rendre visible les solutions architecturales en se dotant des moyens de communications nécessaires et accessibles par tous les acteurs d'un projet.

I.2.4. CONCLUSION DU SOUS CHAPITRE I.2:

L'étude sur la conception architecturale, nous a révélé l'existence de deux visions relatives à la participation des acteurs quant à la production d'un projet architectural.

- Le travail de P.Boudon qui situe les acteurs outre que l'architecte comme participant à une décision dans la production d'un projet en prenant position sur les objets déjà conçus.

- Les autres auteurs, par exemple (Prost, M.Callon, T.Tidafi), qui considèrent que la conception est un travail collectif. Il se réalise par la contribution de tous les acteurs de la conception. Ces acteurs ne dissocient pas la conception et la réalisation d'un projet.

Pour notre part, nous considérons que la tendance situant la conception architecturale comme un travail collectif est plus raisonnable car même l'histoire nous a montré que plusieurs projets d'architecture ont été bâtis grâce à la pratique coopérative de ses acteurs. Faut-il rappeler que beaucoup de projets ont été le fruit de coopération entre les différents métiers et dont le résultat de leur travail reste un succès. Les architectures vernaculaires et classiques sont deux bons exemples de tentatives d'architecture sans architecte ou les constructions sont les résultats de la pratique coopérative de plusieurs acteurs.

L'étude de ce sous chapitre, nous permet de conclure :

Pour que la coopération entre les acteurs de la conception soit effective, nous estimons qu'il y a nécessité d'un outil permettant de figurer les solutions et qui tiennent compte des compétences de tous ces acteurs.

I.3. LES OUTILS DE REPRESENTATION CONVENTIONNELS :

I.3.1. BREF HISTORIQUE SUR LA FIGURATION ARCHITÉCTURALE :

Si l'architecture a existé avant que l'homme ne soit capable de la concevoir dans un projet dessiné, le projet d'architecture a pris naissance le jour où l'homme a trouvé les outils de figurer ses œuvres (P.Quintrand, 1985).

Dans ses dix livres d'architecture, Vitruve considère que l'architecte doit avoir les connaissances du dessin afin qu'il puisse sans effort montrer l'aspect du travail qu'il propose (R.Laurent, 1989). Durant les périodes grecque et romaine, il n'existait pas de dessins d'architecture. Les techniques utilisées sont basées sur la scénographie et l'iconographie. De même, la période du moyen age, n'a pas connu du nouveau en matière de techniques de représentation architecturale.

Dès la période de la renaissance italienne, les travaux de Brunelleschi puis d'Alberti ont ouvert la voie de la conception moderne du projet d'architecture. Cette mise au point faite à la renaissance a permis la séparation entre la conception et la réalisation (C.Parisel 1990).

La représentation par la perspective débouche sur la méthode de projections orthogonales initiées par Raphaël en 1519 et qui furent développées d'une part, par Gaspard Monge avec la géométrie descriptive (1795) ou la notion de géométral s'est imposée comme mode de représentation de l'espace, et d'autre part sur la géométrie projective Poncelet (1822).

Selon R.Laurent (1988), Le géométral est la représentation d'un objet par une projection (orthogonale) sur un plan. En architecture, c'est la représentation d'un bâtiment par son plan ou par son élévation.

I.3.2. LES OUTILS DE DESCRIPTION TRADITIONNELS :

Afin de communiquer les différents aspects de la conception architecturale, il existe plusieurs techniques de figuration que les concepteurs pouvaient utiliser Chaque support de communication renferme des informations qui lui sont propres. La compréhension de l'information représentée souvent de façon conventionnelle, suppose la disposition d'un savoir permettant d'interpréter l'information (Dieu –Hanh Pho 1997).

Parmi ces moyens, on distingue :

I. 3.2.1. REPRÉSENTATION TEXTUELLE :

L'écriture est une forme d'expression couramment utilisée. Les informations textuelles englobent tous les écrits utilisés pour décrire une information relative aux divers sujets. Elles sont caractérisées par un langage spécifique et technique. Elles sont, soit de nature subjective, soit de nature objective (Dieu –Hanh Pho, 1997). Très souvent chaque utilisateur interprète l'information selon son imagination, ce qui peut générer plusieurs interprétations pour un même sujet.

L'inconvénient des représentations textuelles réside d'une part dans l'information que véhicule cette représentation qui est très difficile à visualiser. L'interprétation de chaque image diffère d'un utilisateur à un autre (Dieu – Hanh Pho 1997). D'autre part, l'information que génère la forme textuelle est toujours subjective selon l'auteur.

I. 3.2.2. LA REPRÉSENTATION GRAPHIQUE :

La représentation graphique concerne toutes les autres formes de description utilisées pour décrire une information, il s'agit de graphes, de dessins et des images. Selon le message que l'on veut transmettre, on choisit l'une de ces trois descriptions pour exposer l'information souhaitée.

I. 3.2.2.1. LES GRAPHES :

Afin de communiquer sur les diverses formes de performances du bâtiment, on fait référence aux graphes, où il existe une multitude de modèles symboliques (relations mathématiques) permettant d'évaluer les performances du bâtiment soumis à certains phénomènes (C.Parisel, 1998). Ces informations possèdent un grand niveau d'abstraction. Il est difficile pour un simple usager de comprendre et d'imaginer l'interprétation des résultats. Seuls, les professionnels peuvent comprendre et réutiliser l'information exposée.

I. 3.2.2.2. LES DESSINS :

Les dessins sont utilisés pour codifier et représenter la forme géométrique de l'objet en faisant usage des diverses projections (esquisses, coupes, façades, axonométries, perspectives). L'un des inconvénients du dessin réside dans la description complète d'un objet qui nécessite plusieurs représentations. L'information contenue dans ces représentations est souvent redondante. La compréhension des représentations figurées dans le dessin nécessite une connaissance préalable des significations de certains codes conventionnels qu'utilisent les professionnels. L'autre inconvénient, c'est que les dessins sont toujours accompagnés des devis descriptifs (Kalay,1996) pour apporter des précisions sur certaines caractéristiques supplémentaires, telles que la nature des matériaux, leurs dimensions, leurs quantités, leurs provenances ou leurs performances.

I. 3.2.2.3. LES PHOTOGRAPHIES :

Les photographies sont aussi utilisées pour illustrer certaines informations dont le but de mieux expliquer certains aspects du bâtiment. Elles sont prises de façons figées, au nombre toujours limité, pour montrer un objet ou un détail important que l'on veuille expliquer. Leurs contenus ne donnent qu'une idée limitée de l'objet illustré, ce qui peut constituer une entrave pour comprendre la globalité de cet objet (Dieu – Hanh Pho, 1997). L'autre inconvénient de cet outil réside dans l'impossibilité d'illustrer un objet encore non conçu.

I. 3.2.3. LES MAQUETTES :

Les maquettes sont des représentations volumiques à des échelles réduites mais fidèles dans leurs proportions et leurs aspects que l'on utilise pour communiquer sur un bâtiment ou d'un simple détail architectural.

Si sur le coté visuel, elles sont considérées comme un moyen très explicatif d'un projet, leurs productions permettent difficilement des rectifications s'il y a lieu.

I.3.3. LES LIMITES DES DESCRIPTIONS TRADITIONNELLES :

I.3.3.1. LES LIMITES TECHNIQUES :

C. Parisel (1990) évalue l'ensemble des informations sur le bâtiment à certains critères principaux qui peuvent être des indices d'une bonne conception à savoir :

- **La redondance** : S'évalue en fonction du nombre d'information répétée ou qui existe plus d'une fois. En effet, pour figurer un objet, nous devons faire appel à plusieurs figurations au même temps (plans, coupes, façades …). Très souvent, un même point est représenté sur plusieurs projections (figure n°1).

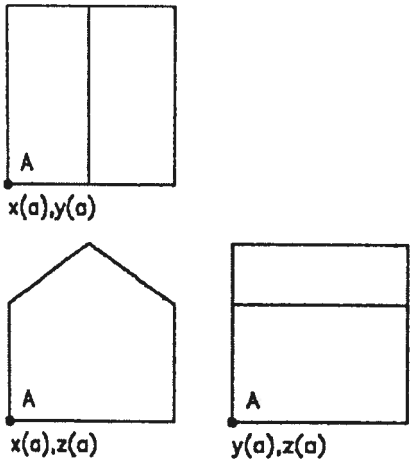


Figure 1.
Les projections orthogonales, un exemple de redondance.

Source : C. Parisel (1990).

Figure n° 01 : Les projections orthogonales : un exemple de redondance.

- **L'ambiguïté** : s'évalue en fonction du nombre de fois, une information sur un élément décrit n'est pas assez complète pour déterminer cet élément. Les informations que présentent les projections orthogonales sont souvent insuffisantes et laissent place à de nombreuses ambiguïtés sur certains détails de l'objet en conception (figure n° 2).

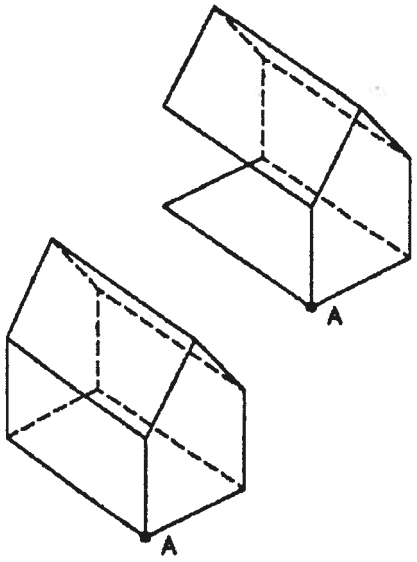


figure 2.
exemple d'ambiguïté dans les projections orthogonales.

Source : C. Parisel (1990).

Figure n° 02 : Les projections orthogonales : un exemple d'ambiguïté.

- **La cohérence** : s'évalue en fonction du nombre de fois que la même unité d'information est répétée. En effet, lorsqu'une information est redondante, toute modification sur une projection engendre une nécessité de mise à jour sur les autres projections (figure n° 3).

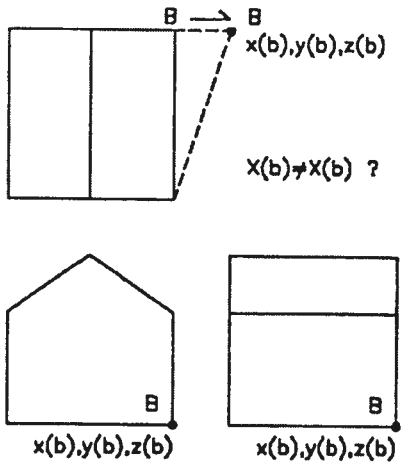


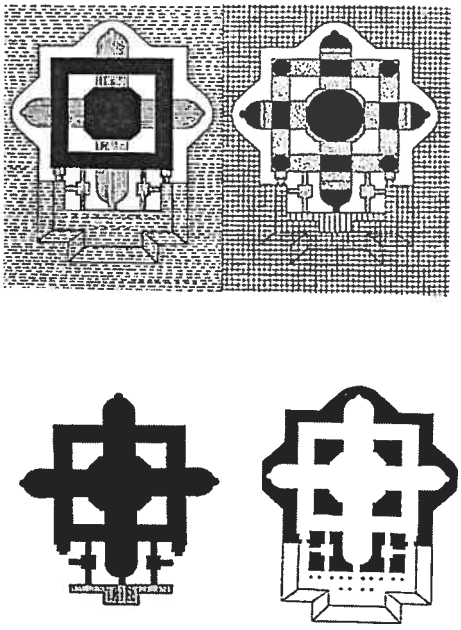
figure 3.
exemple d'incohérence dans les projections orthogonales.
Le déplacement de B en plan n'a pas été répercuté dans l'élévation amenant une incohérence dans la valeur de $X_{(b)}$.

Source : C. Parisel (1990)

Figure n° 03 : Les projections orthogonales : un exemple d'incohérence.

I.3.3.2. LA REUTILISATION LIMITÉE :

L'information représentée par les outils traditionnels prend divers aspects selon les intentions de traitement, même qu'elle s'applique au même sujet. La réutilisation de cette information n'est pas toujours évidente car il est parfois spécifique à un aspect particulier de l'objet représenté (D.H.Pho, 1997) (figure n° 4).



Source : (D.H.Pho, 1997): quelques exemples de Saint Pierre de Rome (Zevi, 1959).

Figure n° 04 : Exemple de divers aspects que peut prendre une information des outils traditionnels.

I.3.3.3. L' INFORMATION NON TEMPORELLE :

Bruno Zevi[1] ; pose le problème de la représentation de l'espace comme problème majeur de la représentation de l'architecture pour la voir, la comprendre et la concevoir. Il constate l'incapacité des outils et des méthodes traditionnelles à représenter la « quatrième dimension ». L'évolution d'un projet à travers le temps n'est jamais représentée. Ces outils offrent seulement la possibilité de figurer des résultats fixes et ponctuels.

---

[1] ; : Quintraud .P/ Autran.J /Floren Zano.M / Fregier.M/Zoller.J (1985) : La CAO en architecture. Paris édition : Hermes.

I.3.3.3. DISCORDANCE ENTRE LES FINS ET LES MOYENS DES ACTEURS :

T.Tidafi (1996) a relevé deux discordances entre les fins des acteurs de processus de communication et les moyens disponibles pour la communication. Ces deux discordances concernent la démarche qui aboutit au résultat qui n'est jamais figurée et la compréhension d'une solution figurée qui diffère d'une personne à une autre selon son psychique, ses connaissances, et ses expériences.

La rigidité des outils traditionnels ne permettent de consigner que le résultat mais la façon dont on a aboutit à ce résultat n'existe jamais. La compréhension de l'information représentée souvent de façon conventionnelle peut différer d'une personne à une autre.

I.3.4. CONCLUSION DU SOUS CHAPITRE I.3 :

A travers ce chapitre, nous retenons que l'utilisation des outils traditionnels comme outils de conception présentent plusieurs limites qui peuvent entraver une bonne coopération entre ces acteurs. La rigidité des outils utilisés n'assure que des résultats fixes et ponctuels. Ils ne peuvent satisfaire la production des solutions produites en permanence.

L'information représentée souvent de façon conventionnelle n'aide seulement les acteurs de conception à matérialiser l'image mentale de cette solution. Cette image mentale est souvent différente d'une personne à une autre. Ce qui peut souvent créer des confusions par rapport à un résultat particulier.

I.4. LA FIGURATION ET L'OUTIL INFORMATIQUE :

I.4.1 INTRODUCTION :

La géométrie plane, outil traditionnel de la figuration de l'espace, présente des points d'incompatibilité avec la projection mentale du concepteur. L'introduction de l'informatique dans le domaine de la figuration architecturale a affecté le dessin graphique en le reléguant à une technique du passé sans retour possible dans l'exercice de cette profession. L'arrivée de l'outil informatique n'a pas été sans incidence. La révolution ayant suivi son adoption a bouleversé cette activité par l'ampleur de ses résultats.

Après un passage d'adaptation des méthodes traditionnelles de figuration, ce nouvel outil prometteur a relancé le débat sur la figuration contemporaine en architecture. Les supports à la production architecturale se " virtualisent " et soulèvent par la même des interrogations sur les modes de production et de communication de la création architecturale. La conquête de nouvelles techniques de figurations et la révision des méthodes de travail de l'architecte suscitent, aujourd'hui, un regain d'intérêt de la part de plusieurs chercheurs.

Afin, de voir comment on peut exploiter l'informatique comme un outil d'aide à la conception architecturale, nous avons commencé par comprendre comment cet outil est utilisé pour figurer une solution architecturale et quelles sont les limites de son exploitation relativement aux attentes des usagers.

I.4.2. LES OUTILS ET PRATIQUES DU DAO (DESSIN ASSISTÉ PAR ORDINATEUR) :

> *Pour les professionnels de la construction de l'espace, designers, architectes, ingénieurs, urbanistes, le dessin est avant tout un outil. Il est le moyen par lequel le concepteur donne forme à ses interlocuteurs. Le dessin a donc pour tâche de donner à voir et à comprendre le projet.*
>
> *(P.Boudon, F.Pousi, 1998)*

Tout comme le dessin traditionnel, le DAO (dessin assisté par ordinateur) est utilisé pour figurer la forme géométrique d'un objet, en faisant usage des diverses projections (plans, coupes, façades, axonométries, perspectives). Plusieurs facteurs ont favorisé le passage de la planche à dessin à l'ordinateur. Parmi ces facteurs, on distingue d'une part, certaines facilités en matière de stockage, de classement, de mémoire et de facilités de reproduction qu'offre l'ordinateur. D'autre part, le DAO permet une grande souplesse et rapidité de réalisation accrue. Il entraîne un gain de temps, allége le travail de l'architecte, favorise la créativité et tout en multipliant la productivité.

Le DAO n'est réellement qu'une adaptation des méthodes traditionnelles aux outils informatiques. Il a succédé au dessin traditionnel essentiellement en ce qui concerne les corrections rapides et les duplications des éléments répétitifs. Si le support de la représentation a changé, la nature de la représentation architecturale est restée la même (R.Coyne, 1996). Elle n'est qu'une adaptation de la forme de description traditionnelle. Les utilisateurs des outils de la DAO doivent connaître les mêmes façons utilisées sur la table à dessin en matière de conventions et de normes pour figurer un objet.

Le principe de base du DAO repose sur la réutilisation de l'information qu'elle soit graphique ou alphanumérique : Tout ce qui est déjà dessiné ou écrit une fois est réutilisable (G. De Paoli et P. Pellissier, 1992). Les inconvénients de ce mode de figuration résident, d'une part, dans la description complète d'un objet qui nécessite plusieurs représentations. On ne peut comprendre un plan qu'en se référant à la coupe. La compréhension des représentations figurées suscite une connaissance préalable des significations des codes conventionnels utilisés.

I.4.3. CONCLUSION DU SOUS CHAPITRE 1.4 :

Même si Le DAO est la version informatique moderne du dessin traditionnel et qu'il offre tous les avantages de l'ordinateur (stockage, classement, mémoire et facilité de reproduction), les formes utilisées pour figurer un objet n'ont pas évolué. On utilise les même projections (plans coupes, façades …) héritées du dessin traditionnel. Nous devons connaître les techniques et les conventions utilisées en architecture.

C'est pourquoi les facteurs qui freinaient les méthodes traditionnelles de conception à représenter l'architecture dans sa totalité demeurent non résolus aussi avec le DAO.

I.5. LA FIGURATION ET LA CAO (CONCEPTION ASSISTÉE PAR ORDINATEUR :

I.5.1. INTRODUCTION :

La conception assistée par ordinateur (CAO) est généralement considérée comme une technique permettant de modéliser un objet à concevoir, de l'évaluer, de l'affiner et d'opérer les modifications nécessaires avant de le réaliser. C'est une technique de simulation qui offre la possibilité d'explorer un nombre important d'hypothèses. Cependant, son utilisation dans les premières phases de conception architecturale n'est pas effective. L'architecte n'utilise la CAO que pour produire des formes architecturales au lieu d'un véritable outil d'aide à la conception. L'outil informatique n'est pas utilisé comme outil d'aide à la conception mais plutôt comme outil d'aide à la modélisation.

Très souvent, l'apport de l'informatique dans le domaine de l'architecture est limité à la production de l'image. Il repose principalement sur le concept de l'observation. Cette notion de limiter l'outil informatique aux seules utilités graphiques est héritée de la rigidité des outils traditionnels.

Violet Le Duc[1] ; affirme dans son article du dictionnaire relatif à l'ogive : " dessiner c'est voir, et voir c'est savoir". Cette phrase lui a valu beaucoup de critiques notamment de (P.Boudon 1992)

En effet en proposant un modèle intelligible de l'ogive gothique, Violet Le Duc explique qu'il n'est pas seulement une figure géométrique mais note que son importance réside dans son utilisation en tant que système constructif économique et pratique pour le chantier.

---

[1] ; tiré de (P.Boudon 1992)

Boudon pense que sa théorie "voir c'est savoir" ne s'applique pas nécessairement à son travail. Dans son modèle, il explique comment par l'ogive qui permet des claveaux taillés de manière différente, on pourra faire des modèles d'arcs de largeurs différentes :

> *"Avec la même ouverture de compas, nous pouvons avoir une suite d'arcs dont les diamètres seront au diamètre du plein-cintre, qui est le plus grand arc de voûte de l'arc ogive.... Ayant donc des claveaux tous taillés dans un même arc, nous pouvons, sans épure, monter tous les arcs d'un édifice".*

Par cet exemple même de l'ogive de Viollet-le-Duc, Boudon explique bien que «voir n'est pas savoir» et que l'image de synthèse ne saurait répondre à la complexité de la conception. Il conclut que l'immersion dans le visible par l'informatique réduit la conception au dessin.

Par cette critique de P.Boudon, nous devons comprendre que la conception d'un objet architectural n'est pas seulement sa figuration mais surtout son processus de création. Ainsi, pour conformer l'informatique à un outil d'aide à la conception, nous devons exploiter le potentiel informatique en tant qu'outil d'assistance à la conception au lieu d'une simple commodité d'exécution destinée aux fonctions techniques. Aujourd'hui, il est utilisé que pour des fonctions nécessitantes des précisions. Il est resté seulement utile à une courte phase du processus global de conception : création, visualisation, manipulation formelle du modèle géométrique (G. De Paoli ,1999).

I.5.2. PROPOSITIONS D'EXPLORATIONS UTILISÉES EN CAO :

Afin de mieux comprendre le sens de la CAO, plusieurs chercheurs ont tenté de trouver des formules pour exploiter l'outil informatique dans sa véritable mission d'aide à la conception. Pour cela, nous avons commencé par explorer quelques approches proposées par certains chercheurs dans ce domaine:

Depuis l'appropriation de l'outil informatique dans le domaine de la conception architecturale, plusieurs tentatives d'exploration ont été essayées. Parmi les thématiques investies en architecture par la CAO : Les grammaires de formes proposées par Stiny (1980) qui consistent à appliquer des règles de réécritures sur les formes géométriques tel que l'on peut le faire sur des chaînes de caractère. Le principe est d'assimiler les éléments à concevoir à des mots de vocabulaires alors que les règles de composition à des règles de grammaire. Cette approche peut être appliquée à l'analyse d'architecture existante mais aussi pour la génération des objets. Selon Stiny (1993), des problèmes existent dans le contrôle de l'exploration et dans la recherche d'appariements des règles sur des formes implicites.

Flemming (1987) de l'université Carnegie Melon a proposé quant à lui, un système de génération automatique de solutions architecturales. En se basant sur deux règles préétablies (organisations et articulations) et après l'étude faite sur des modèles existants sur le style Queen Anne, le système pouvait générer de nouveaux modèles de maisons de n'importe quel style.

Le système proposé par Flemming éprouve plusieurs limites dans la mesure où le système ne permet pas d'interaction entre l'utilisateur et le système (génération automatique). Ce système n'offre pas vraiment le choix, il accorde à l'utilisateur des solutions qu'il doit accepter ou non, ou bien des solutions irrecevables.

L'autre thématique investie par la CAO, *les* travaux visant à formaliser les connaissances architecturales de Coyne R. D., Rosenman M. A. & al. (1990). Le principe est de traduire les connaissances en termes de modèles ou schémas génériques, à partir desquels un système peut assister un concepteur Coyne & al. (1990). Si l'assistance à la conception par les schémas est intéressante, cependant, elle présente certaines limites notamment en matière de masse de connaissances F.Guena (1997).

Plusieurs chercheurs ont critiqué ces approches relatives aux systèmes à base de connaissances appliquées à la conception architecturale. Ces chercheurs notent que les techniques utilisées, en

raisonnant sur des schémas décrivant des solutions générées où des cas décrivant des solutions particulières peuvent être regroupées dans le terme général des réutilisations de précédents. Ils proposent contrairement à ces systèmes complètement automatiques de réutilisations des précédents, une autre voix permettant la réutilisation des précédents, tout en laissant le choix de conception aux utilisateurs F.Guena (1997).

Avec la performance des moyens informatiques, certains outils de CAO commencent à intégrer les intentions du concepteur. La tendance d'évolution des systèmes de CAO est intégrée au modèle purement géométrique d'autres informations non géométriques et à des niveaux différents. L'objectif est de permettre une mise à jour du modèle après une modification d'une partie de l'objet. La conception d'un objet est un raffinement continuel, c'est une boucle de conception/ reconception.

Plusieurs approches existent pour répondre à ces besoins notamment les approches paramétriques. Parmi ces approches, (W.J.Mitchell, 1991) propose une approche basée sur des paramètres fonctionnels et procéduraux. Cette approche permet de consigner les procédures de création de l'objet.

Afin d'assister les acteurs de la conception architecturale dans les pratiques de coopération, certaines approches actuelles tendent à développer des outils tenant compte de l'activité de co-conception (Catherine Deshayes 2003), (Peter Fröst, 2003) (Achten, H.H 2002). Parmi ces approches, il existe celles qui permettent à certains acteurs de la conception notamment les clients à accéder à un savoir graphique. Grâce à la visualisation possible en 2D et 3D, ces outils aident à faciliter la représentation mentale, à favoriser la communication et à permettre une réelle co-conception. Ils permettent un passage de l'idée du client à une forme visible et exploitable.

Ces logiciels permettent aux clients de présenter d'une nouvelle manière leurs demandes. Ils bouleversent la relation entre un client et un architecte mais ne gêne en rien l'activité de

conception spatiale de l'architecte. Grâce à la représentation informatique produite, la nouvelle relation entre le client et l' architecte, favorise un travail de conception coopérative, de support de communication et assure aux clients de mieux suivre l'évolution de cette conception (Catherine Deshayes 2003),

Dans la même perspective de doter la communauté de professionnels du bâtiment d'un environnement logiciel de travail collaboratif, l'équipe : Jean-Claude Bignon, Gilles Halin, Damien Hanser, Olivier Malcurat, (2002) propose une nouvelle approche de COCAO : Modélisation d'un environnement logiciel coopératif pour les acteurs de l'architecture et du B.T.P (bâtiment et travaux publics). Ce projet concerne surtout la structuration de l'information, l'organisation des échanges et l'accessibilité des services. Même si l'expérimentation semble apporter une réponse au besoin de mobilité des acteurs, le système utilisé de représentation des documents en dossiers et sous dossier (arborescence) ne permet pas de representer de manière efficace la complexité des relations entretenues entre acteurs et documents.

A travers le survol de ces approches que nous venons d'étudier, nous retenons que certaines approches tiennent compte de l'activité coopérative de la conception architecturale et permettent d'intégrer les intentions des acteurs de la conception architecturale.

I.6 L'OUTIL INFORMATIQUE ET LES ACTEURS DE LA CONCEPTION ARCHITECTURALE :

I.6.1 INTRODUCTION :

> *Un des effets immédiats de l'application des ordinateurs à la conception en architecture sera de modifier les interactions entre les participants habituels. En rendant le processus de design plus explicite, plus publique et plus systématique. Les ordinateurs auront l'effet inévitable de permettre la collaboration effective d'un grand nombre de cerveau.*
>
> (Lichnerowicz , 1971)

Chaque figuration qu'elle soit matérielle ou graphique, véhicule un savoir mais possède également ses propres limites. Elle peut assurer ou interdire certaines interactions entre partenaires. Si on cite le cas de la maquette qui, grâce à la variation d'échelle, on peut faire une description volumique d'un objet architectural, le choix de ce mode de figuration assure l'interaction entre les différents partenaires, mais il ne permet pas à un acteur d'exprimer son point de vue.

L'arrivée de l'ordinateur dans le domaine de l'architecture a relancé la question de participation des usagers dans le processus de conception architecturale. L'appropriation de l'outil informatique a suscité au début, des questionnements (Lichnerowicz, 1971), comme par exemple si l'utilisation de l'ordinateur ne répondait pas seulement au soucis d'économiser la main-d'œuvre que pour améliorer la qualité du processus de design.

L'introduction l'informatique dans tous les domaines a eu pour effet, des changements techniques qui ont touché les rôles et les relations humaines. En architecture, un des effets immédiats de l'application des ordinateurs à la conception architecturale a été la modification

des interactions entre les participants. L'ordinateur a redéfini la tâche de collaboration entre les divers participants et a rendu le processus de design plus explicite (Lichnerowicz, 1971).

Les utilisateurs des bâtiments sont habituellement exclus du processus de conception. Avec l'introduction de l'outil informatique dans le domaine la conception architecturale, plusieurs questions sur les perspectives d'élargissement de l'éventail des participants au processus de conception et du rôle à jouer par l'utilisateur pour satisfaire ses besoins commençaient à être posées (Lichnerowicz, 1971).

L'hypothèse voulant que la possibilité d'introduire l'outil informatique à la conception architecturale permettrait aux usagers et autres acteurs participant à un projet d'architecture de prendre part à toutes les étapes du processus de conception a ouvert la piste à plusieurs chercheurs. En effet, dès les premières années de son introduction, tel que mentionné par (Lichnerowicz 1971), Maver a proposé, durant l'année 1970, un mécanisme pour la participation des acteurs de la conception, en suggérant de créer une ''équipe de résolution'' composée de clients, usagers, financiers et représentants de la société ...etc. L'ordinateur évaluerait d'abord les propositions de l'équipe de conception. Après, ces propositions et solutions seraient soumises à cette équipe de résolution qui voterait sur l'alternative à retenir.

Selon M. Callon (1996), au début de l'appropriation de l'outil informatique par les architectes, les travaux de son intégration, se sont heurtés au manque de maturité des méthodologies, de structuration de données et à la faiblesse des environnements informatiques (langages et machines). Aujourd'hui, les chercheurs dans le domaine lié à l'amélioration des outils d'aide à la conception par ordinateur essaient d'intégrer les recherches en amont relatives à la modélisation des objets architecturaux. Les systèmes recherchés permettraient une exploitation dès les premières phases de conception en offrant aux usagers de ces outils des automates pour développer leurs propres stratégies de travail. Il s'agit en terme informatique, d'améliorer les interactions homme machine (partage de connaissances, manipulation graphique et sémantique).

Aujourd'hui, avec la performance des outils informatiques, plusieurs chercheurs abordent la question de comment les métiers de l'architecture et de la construction peuvent tirer des avantages et des bénéfices des outils informatiques dans les activités coopératives.

I.6.2. ÉTUDE CRITIQUE D'UN MÉDIUM D'AIDE À LA COMMUNICATION ARCHITECTURALE :

En abordant l'utilisation de l'ordinateur comme médium de communication entre les acteurs collaborant dans un projet d'architecture (ingénieurs, gouvernement, clients ...) Tsuyoski Sasada [1] (1995) de l'université d'Osaka a identifié deux carences liées à l'utilisation de ses outils pour des fins de communications :

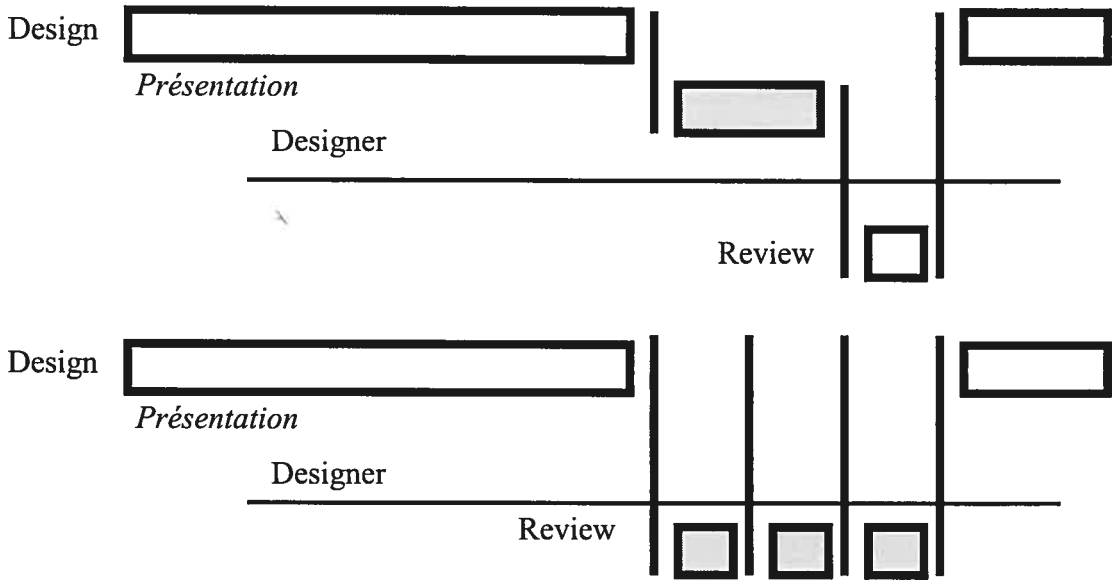
Le premier problème est celui de l'initiative. En voulant représenter et communiquer leurs travaux, beaucoup de professionnels (architectes ou designers créent des animations ne communiquent que ce que ces professionnels veulent rendre visible.

- Le premier problème est celui de l'initiative. En voulant représenter et communiquer leurs travaux, beaucoup de professionnels (architectes ou designers créent des animations. Très souvent, ces animations ne communiquent que ce que ces professionnels veulent rendre visible. Les autres acteurs, destinataires de ces animations ne peuvent pas observer une autre partie de ce projet autre que celle contenue dans les animations présentées.

- Le deuxième problème auquel est confrontée la communication d'une solution architecturale par ordinateur est le problème de ''timing'' : certains modeleurs 3D actuels offrent des modèles fixes où il est difficile de reproduire d'autres alternatives, si les participants constatent certains problèmes ou ne partagent pas l'avis du concepteur (figure n°5).

---

[1] : *CAAD Futures: Computer Aided Architectural Design Futures*



Source : *Computer Aided Architectural Design Futures, 1995*

Figure n° 05 : Reproduction d'alternatives

L'analyse que l'on peut tirer de ce médium de communication présenté par Tsuyoski Sasada est que cet outil facilite la correction en permettant de reproduire de nouvelles solutions architecturales en fonction de ce que souhaitent les autres participants mais les autres acteurs, destinataires de ces animations ne peuvent pas apporter des changements sur les modèles car ces changements à apporter nécessitent une connaissance du langage de programmation utilisé. C'est pourquoi que l'on peut dire que même avec ce medium, les acteurs de conception qui ne connaissent pas le langage de programmation utilisé seront considérés comme des participants à la décision que des véritables acteurs de la conception.

I.6.3. TYPES D'INTERACTIONS QUE PRESENTENT LES OUTILS DE CAO :

Selon I. Iordanova (2000), l'interaction avec un modèle s'effectue de deux façons :

Une interaction graphique directe sur l'écran. Les modeleurs utilisant cette forme d'interaction (Autocad, Euclid, Catia, Form_Z...) sont plus des logiciels d'aide à la production de modèles. Grâce à leurs possibilités de résoudre certaines contraintes en intégrant certaines fonctions permettant de figurer quelques formes géométriques complexes, ces derniers assurent seulement à produire des maquettes numériques d'un objet. Ils sont donc utilisés que pour figurer et communiquer une information relative à l'aspect géométrique de l'objet.

Le deuxième type d'interaction est celui qui se base sur les différentes sortes de programmation (déclarative, impérative, orientée objet, ou fonctionnelle). Les modeleurs utilisant ce type d'interaction, permettent, en plus de produire des maquettes numériques, de laisser à la machine de générer des volumes selon des règles préétablies.

Selon N. Charbonneau 2002, plusieurs chercheurs (Krawczyk, 1997, Darlymple& Gerzso 1998, Piazzalunga & Fitzhorn, 1998) ont démontré que les systèmes se basant sur les langages de programmation sont beaucoup plus puissants relativement aux outils basés sur des interactions graphiques pour construire un vocabulaire ou formuler les règles.

I.6.4. CONCLUSION DU CHAPITRE 1.6 :

A travers ce chapitre, nous constatons que malgré la possibilité des outils de CAO à rendre visible les solutions architecturales par une médiation visuelle produite en permanence entre les différents partenaires, Les mediums assurant le dialogue entre l'outil informatique et les acteurs de la conception architecturale ne répondent pas aux besoins de tous ces derniers.

A travers, la prospection des interactions que présentent les outils actuels de CAO, nous constatons que la création d'un assistant manipulable par tous les acteurs passe par l'instrumentation du dialogue homme machine, tout en profitant de la puissance des modeleurs se basant sur les langages de programmation.

Ainsi, pour rendre manipulable les outils de CAO par les acteurs de la conception architecturale, nous devons créer les conditions de dialogue entre ces acteurs et l'outil informatique.

I.7) : CONCLUSION DU CHAPITRE I :

A travers notre recherche bibliographique, nous avons dégagé un certain nombre de disproportions entre les connaissances théoriques de la conception architecturale et les outils mis à la disposition des concepteurs.

La conception architecturale est une activité collective faisant appel à tous les acteurs d'un projet d'architecture. Très souvent, ces acteurs sont issus de différents niveaux d'instructions. Les outils actuels d'aide à la conception présentent certaines discordances par rapport aux besoins des concepteurs.

A travers notre recherche nous avons conclu qu'un outil d'aide à la conception architecturale doit répondre à certaines conditions :

- Permettre la possibilité de rendre visible les solutions architecturales par une médiation visualisation produite en permanence entre les différents partenaires.

- Faciliter la manipulation par tous les acteurs participants au processus de conception architecturale quelques soient leurs niveaux d'instructions Michel Callon, (1996).

- Surpasser la notion de '' timing'' tel que expliqué par Tsuyoski Saada (1995), c'est-à-dire qu'un utilisateur puisse reproduire de nouvelles solutions architecturales en fonction de ce qu'il souhaite sans pour autant être obliger de réécrire à nouveau sa solution.

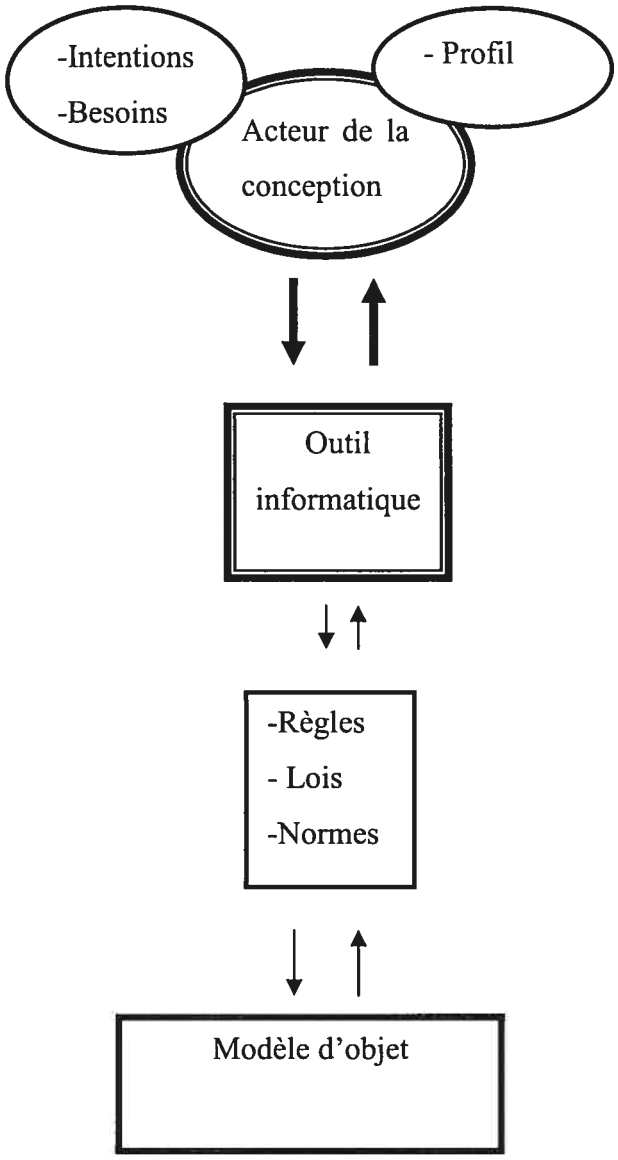


Schéma de fonctionnement de l'assistant envisagé

II. MÉTHODOLOGIE

Selon A-P.Contandriopoulus (1990), durant cette phase, notre tâche se résumerait à faire un bilan des avantages et des inconvénients des diverses stratégies de recherche connues. Ce travail nous permettra de décrire la stratégie à adopter pour répondre à la question de la recherche et de vérifier les hypothèses spécifiées. Nous devons alors soumettre les hypothèses et les questions à l'épreuve des faits.

Pour ce faire, il existe deux caractères qui permettent d'évaluer la qualité d'une stratégie de recherche retenue et de soumettre l'hypothèse à l'épreuve des faits. Il s'agit de la validité interne et la validité externe (A-P.Contandriopoulos 1990).

La validité interne est avalisée par les caractéristiques du devis qui assure que les relations retenues entre les variables ne peuvent être expliquées avec d'autres facteurs. Nous devons éliminer toute autre explication que celle retenue dans notre devis.

La validité externe dépend du caractère plus au moins général du modèle théorique sur le quel se base un travail de recherche. Il faut prouver que les résultats obtenus peuvent être généralisé à tout autre circonstance.

Ainsi dans le même ordre d'idées, pour répondre à la question de notre recherche et vérifier nos hypothèses, nous avons commencé par faire un survol des stratégies existantes ou nous avons à éclaircir les points forts de chacune d'elles. Ce travail nous amènera à dégager une stratégie appropriée à utiliser pour apporter les éléments de réponse à notre questionnement.

Une fois notre stratégie connue, nous devons vérifier nos hypothèses en choisissant un cas d'étude (l'escalier dans notre cas) tout en prouvant que les résultats obtenus peuvent être généralisés à d'autres cas similaires.

II.1 LES CONDITIONS DE LA FIGURATION DES BESOINS DES ACTEURS :

Afin de formuler une solution, un acteur de la conception architecturale doit identifier les éléments de son problème. La solution choisie doit être une réponse satisfaisante, soit à des buts à satisfaire, des contraintes à prendre en considération, ou des performances spécifiques à atteindre Milne [1].

Si la rigidité des outils traditionnels propose des solutions ponctuelles où il est impossible de consigner toutes les solutions envisageables, aujourd'hui, les outils actuels peuvent offrir d'autres alternatives. Ainsi, si on procède par consigner les buts, les contraintes ou les performances à atteindre, ces nouveaux outils peuvent venir en aide à ces acteurs.

Selon T.Tidafi (1996), pour qu'un mode de figuration réponde aux besoins des acteurs de conception architecturale, il doit satisfaire les deux conditions complémentaires qui sont à prendre en considération à savoir la perception téléologique de tout acteur participant à cette solution et le cours d'une solution architecturale.

C'est en envisageant la perception visuelle comme téléologique qu'un acteur de conception pourrait réaliser ses finalités dans un projet. Tout acteur de conception doit focaliser sur un certain nombre de caractéristiques d'une solution architecturale et distinguer celles qui lui permettraient de réaliser ses actions (T.Tidafi 1996). La satisfaction de ces deux conditions, nous renvoie à construire tant de figures qu'il y a d'acteurs ou de transformations envisageables à une solution architecturale (T.Tidafi 1996).

---

[1] ; : tiré du livre de W.J.Mitchell (1987)

Si les solutions figées et ponctuelles que proposent les modes de figurations traditionnelles ne peuvent satisfaire cette condition, comment doit-on procéder pour rendre visible les différentes possibilités de figurations ?

II.2) : LA MODÉLISATION COMME MOYEN DE FIGURATION DES SOLUTIONS ARCHITECTURALES :

II.2.1. LA NOTION DU MODÉLE :

> *Tout modèle a pour objet de simuler le comportement d'un système en fonction de certains objectifs et compte tenu de certains moyens.*
>
> (B.Walliser, 1977)

Comme dans la plus part des domaines, la réalisation d'une solution architecturale est souvent précédée par sa figuration. L'usage de la figuration, voire même de la simulation des objets est un procédé bien vieux dans l'histoire de l'humanité. Cette pratique qui permet la compréhension est sans aucun doute bien antérieure à l'ordinateur H.A.Simon (1974).

L'arrivée de l'outil informatique dans le domaine de la conception architecturale s'est vite accompagnée par des bouleversements qui ont touché la profession. Avec le passage de l'objet à sa représentation dans l'ordinateur, la modélisation permet la construction d'une véritable maquette virtuelle. Sa qualité réside dans sa capacité à représenter le plus fidèlement possible l'objet réel (J.P. Couwenbergh 1998). Cet avantage lui confère, aujourd'hui, le moyen le plus approprié grâce à la possibilité de pouvoir soumettre l'objet à la simulation des phénomènes qu'il puisse subir.

Selon Le Moigne (1990) :

Modéliser, c'est à la fois identifier et formuler quelques problèmes en construisant des énoncés et chercher à résoudre ces problèmes en raisonnant avec des simulations. Modélisation et simulation, réflexion et raisonnement sont les deux faces inséparables de toute délibération.

Même, si le modèle est une représentation de la réalité d'un objet ou d'un phénomène, cela peut être insuffisant pour répondre aux besoins des acteurs. En effet, si on procède par décrire les solutions envisageables au lieu des processus de leurs créations, il sera difficile de surpasser les descriptions figées et ponctuelles et on ne pourra jamais rendre visible toutes les solutions envisageables.

II.2.2. LA MODÉLISATION PROCÉDURALE :

Aujourd'hui, grâce à l'outil informatique, la modélisation dépasse la simple simulation ponctuelle et figée. Elle permet, en plus la possibilité de figurer le plus fidèlement un objet réel, d'étudier un processus, un système et des démarches cognitives. La notion de modèle fait appel à d'autres façons de décrire des objets en consignant les processus de leurs créations.

C.Parisel et T.Tidafi (1999), notent que la modélisation permettant de visualiser un objet est une activité courante et l'outil informatique peut servir à de nouvelles formes de lecture et de raisonnement. Pour cela, ils proposent dans leur article - le modèle en architecture dans le contexte informatique – la possibilité de l'intégration de la connaissance (metamodéle). Ils pensent à faire des générateurs d'objets stables, en décrivant non pas les objets, mais plutôt chercher leurs processus de création, c'est à dire toute action subie ou provoquée par ces objets et pouvant se répercuter sur leurs formes ou leurs natures. Il peut exister plusieurs types de metamodéles, notamment des metamodéles procéduraux, géométriques, histoires ou de gestion d'un bâtiment.

Dans la même perspective, G.De Paoli (2000) note dans sa thèse de doctorat qu'avec les outils actuels de figuration, il est possible de dépasser cette notion limitant leurs taches à produire seulement des modèles. Pour cela, il propose, une voie de recherche basée sur « la projetation ». Il s'agit de représenter le bâtiment par des fonctions paramétrables qui permettent de créer des maquettes procédurales d'aide à la conception. Cette procédure permettra de créer une famille de modèles partageant certaines propriétés.

Ces deux approches citées précédemment, nous éclaircissent la façon dont nous devons procéder pour aboutir à répondre aux besoins de ces acteurs.

II.3. APPROCHE PROPOSÉE :

II.3.1. INTRODUCTION :

L'assistant que nous proposons, a pour but de permettre aux acteurs de la conception architecturale de naviguer à travers les caractéristiques d'une famille d'objets sans avoir une grande connaissance des outils de CAO.

Afin de répondre à la double question de comment figurer et faciliter le dialogue entre l'homme et la machine, notre approche sera basée sur l'instrumentation des procédures de création des objets. Pour y faire, nous comptons assembler deux approches complémentaires. Cet assemblage permettra d'une part de surpasser les limites de figuration des outils traditionnels, et d'autre part, d'assurer l'interaction entre l'homme et sa machine de la façon la plus naturelle. Cette méthode que nous envisageons, sera composée de la méthode procédurale et de la méthode de dialogue visuel. Chacune des deux méthodes est censée apporter une partie de réponse à notre problématique et leur complémentarité nous permettra de vérifier nos hypothèses.

II.3.2. APPROCHE PROCÉDURALE :

Notre approche découle de la possibilité de rendre visible tout modèle appartenant à une famille d'objet partageant un ensemble de caractéristiques. Aujourd'hui, il existe plusieurs approches qui nous permettent de surpasser la description figée et ponctuelle d'un objet. Ces approches consistent à coder les opérations d'intervention qui produisent les résultats.

Dans sa thèse de doctorat, G. De Paoli a introduit la notion de maquette procédurale qui est un exemple d'intégration des propriétés de l'objet aux primitives géométriques. En analysant le théâtre de Vitruve, (G. De Paoli 1999) a expliqué que l'apport scientifique de cette expérience est la compréhension et la validation de l'hypothèse voulant que nous pouvons intervenir sur des opérateurs qui permettent à l'architecte de se réapproprier une conception complexe du bâtiment.

Si nous choisissons les outils, on peut créer une fonction qui évolue et qui nous permette de créer de nouveaux modèles. Ils pourront être représentés d'une manière complètement différente du modèle de départ, tout en gardant les mêmes propriétés (G. De Paoli 1999). L'exemple de la maquette procédurale du théâtre romain de Vitruve nous montre comment par un modèle générique, nous pouvons naviguer à travers les propriétés de l'objet à concevoir et produire un modèle avec les caractéristiques désirées.

Notre stratégie sera basée sur la notion de processus. L'approche procédurale permet l'insertion d'un changement à venir à travers le temps. Selon Le Moigne (1990), pour que la caractéristique d'une action ou d'une fonction se fasse récursivement : elle passe commodément par la notion générale de processus. Cette stratégie permet une construction directe en 3D, tout en évitant les problèmes de redondance, d'incohérence, d'inconsistance ou de mise à jour. Le modèle résultant est un modèle unique.

II.3.3. APPROCHE DU DIALOGUE VISUEL :

Avant les années 80, les ordinateurs étaient surtout destinés aux spécialistes, c'est pourquoi les fabricants des outils informatiques ne se souciaient pas réellement du problème posé par la vision et la logique d'un système d'exploitation. Avec le développement informatique et son intégration dans tous les domaines, on commençait par prendre le facteur ergonomique en compte, on s'intéresse à la manière dont l'utilisateur appréhende, assimile l'information et au soucis de gain de production, de rendement et du temps.

A fin de rendre accessible l'outil informatique dans les domaines, la question de comment rendre accessible la programmation est toujours d'actualité. Aujourd'hui, les interfaces hommes machines peuvent apporter les éléments de réponse en prenant en compte des aptitudes de chacun des utilisateurs de l'outil informatique.

Avec l'arrivée des systèmes d'exploitation à interfaces graphiques, le monde de la programmation a connu de profonds bouleversements. Plusieurs types de logiciels utilisant des interfaces visuelles ont été développés.

Notre outil devra assurer aux utilisateurs de naviguer et d'interagir avec leurs informations grâce à un langage visuel c'est à dire un langage qui utilise un vocabulaire visuel, qui assure un dialogue et une interaction visuelle et qui peut aussi être défini, tel un ensemble sémantique que l'on utilise pour effectuer des actions de communications avec l'ordinateur.

II.4. DESCRIPTION DE L'ASSISTANT :

II.4.1. INTRODUCTION :

Pour assister efficacement un acteur de conception dans ses tâches quotidiennes, il est nécessaire de représenter les connaissances liées à ses tâches, selon un point de vue qu'il soit le

plus familier. Tout usager de l'outil informatique souhaite, en effet, retrouver au mieux, une adéquation du système d'assistance avec ses compétences.

Ainsi, nous avons décidé dans un premier temps, de voir comment développer un outil capable d'intervenir sur un programme et de lui apporter des modifications, en partant du principe que l'aspect visuel facilite la lecture.

II.4.2. LES AVANTAGES DU LANGAGE VISUEL UTILISÉ :

L'utilisation d'un langage visuel comme langage intermédiaire entre le modeleur et son utilisateur offre plusieurs avantages :

Pour résoudre tout problème, il faut disposer d'un outil fiable et valide de mesure. Selon (A.P.Contiandriopoulos, 1990), la fiabilité est la qualité d'un instrument à mesurer fidèlement un phénomène. La validité est la capacité d'un outil à bien mesurer le phénomène à l'étude.

Si la notion de fiabilité est une caractéristique de l'outil lui même, indépendante de la question de recherche, la notion de validité dépend du contexte d'utilisation de l'outil. Si un outil est valide pour une recherche, il peut être contesté pour une autre.

La construction d'un modèle informatique suppose l'existence d'un langage symbolique comme moyen de décrire un savoir et d'un système géométrique comme moyen de rendre visible ce savoir (T.Tidafi, 1996). En informatique, un langage symbolique est un langage de programmation. Le choix de ce langage est souvent dicté par les finalités de la modélisation.

Dans le domaine de la CAO, il existe plusieurs modeleurs qui sont mis à la disposition des concepteurs, où chaque outil présente des avantages et des limites quant au but visé par la

modélisation. Si un outil est valable pour un but de modélisation, il peut être contesté pour un autre. Pour chaque but recherché, nous devons choisir l'outil le plus approprié. Pour pouvoir communiquer sur tous les problèmes relatifs à la conception, les acteurs de la conception doivent connaître plusieurs modeleurs. Ce qui peut constituer un inconvénient pour eux dans la mesure où ils seront restreints de connaître plusieurs langages de programmation.

Afin de simplifier la tache aux concepteurs et de surpasser la multitude de langage de programmation, l'assistant que nous proposons pourra servir d'intermédiaire entre un concepteur et son ordinateur.

II.4.3. ERGONOMIE DE LA COMMUNICATION :

Tout logiciel a pour but de servir un usager dans ses tâches quotidiennes. Selon C.Ratier (2000), pour dire qu'un logiciel est ergonomique, il doit répondre à deux critères :

- Critères " d'utilisabilité " ou maniabilité : les applications qu'il contient doivent être facile à manipuler et bien adaptées aux profils des utilisateurs ciblés.

- Critères d'utilité : les applications que contient le logiciel doivent être en adéquation avec les tâches des usagers.

Le dialogue entre le concepteur et sa machine, que nous aspirons atteindre doit répondre aux deux critères ergonomiques cités plus haut et aux profils des concepteurs. Afin que notre assistant assure une communication homme machine aussi naturelle que possible. Les taches que le concepteur puisse effectuer, doivent appartenir à un langage correspondant bien à ses attentes, capacités et expériences.

L'image, grâce à ses nombreux avantages : l'assimilation rapide, la stimulation créative, la génération d'idée, la richesse informationnelle, est utilisée comme nouveau support à la

recherche d'informations techniques. Ainsi, par exemple pour mettre en adéquation un acteur de la conception avec la recherche d'informations techniques, l'équipe Halin Gilles, Bignon Jean-Claude, Nakapan Walaiporn, Humbert Pascal, Wagner Marc (2004) a présenté un assistant appelé "batimage" qui se base sur le principe de l'utilisation de l'image comme support à la recherche d'informations.

Cet outil doté d'un robot spécialisé permet à un acteur de la conception de naviguer à travers des images extraites des sites des fournisseurs de produits. En exprimant ses besoins, il peut retrouver des produits et matériaux du bâtiment sans pour autant manipuler le vocabulaire précis du domaine. Si cet outil offre un environnement interactif de dialogue sans manipulation du langage technique. Les résultats obtenus se limitent à des produits de fournisseurs et ne sont réalisables que si les images vérifient trois principes (analogie, forme, contexte).

Contrairement à cet outil, notre démarche consiste à utiliser l'image que dans le principe de l'analogie. Selon E.Morin, (1986) la connaissance par analogie est une connaissance du semblable qui utilise et produit des similitudes de façon à identifier les objets ou les phénomènes qu'elle perçoit ou conçoit.

II.4.4. ASPECT TECHNIQUE DE L'ASSISTANT:

II.4.4.1. INTRODUCTION :

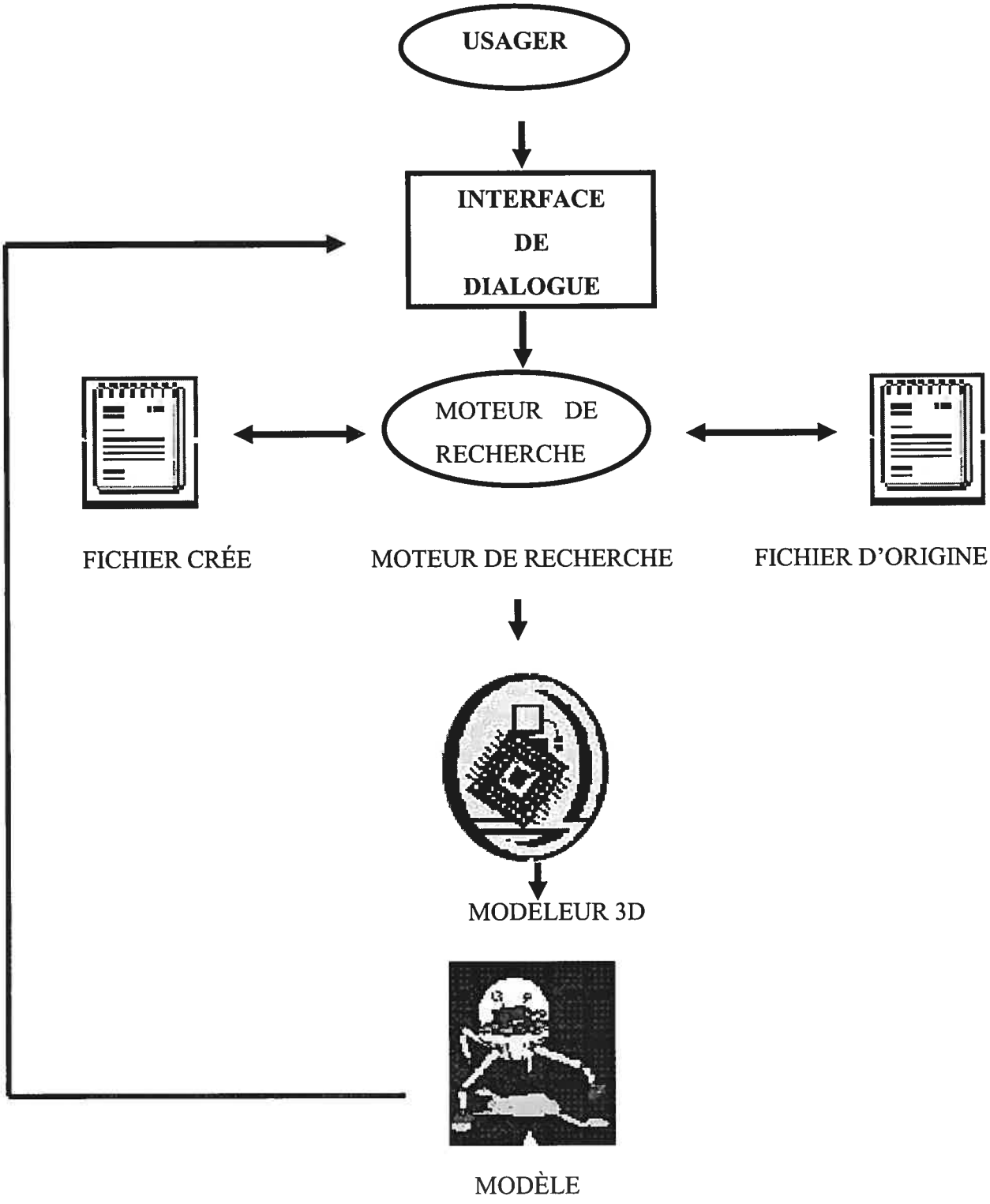
Afin d'utiliser l'informatique par tous les usagers de façon « productive », les producteurs de logiciels ont opté pour l'utilisation des logiciels intuitifs. On a donc surpassé tous les logiciels qui utilisaient des interfaces en mode texte ou en lignes de commande (M.Goutelle, 1997).

Pour rendre le dialogue entre un acteur de conception et son ordinateur aussi naturel que possible, nous comptons créer des interfaces graphiques. Une interface graphique est une interface entre un usager de l'outil informatique et une application. L'interface va donc devoir réagir à des actions de l'usager et aussi à des instructions données par le programme.

Une interface graphique est basée sur 2 grandes composantes différentes :

- L'aspect visuel qui comprend les cadres et les composants d'interfaces qui sont placées dessus.
- L'interaction avec les utilisateurs (action de la souris, du clavier). Il s'agit en fait de la gestion des événements.

II.4.4.2. SCHÉMA DE LA MÉTHODE D'INTERFACE GRAPHIQUE :



II.4.4.3. FONCTIONNEMENT :

Le fonctionnement de notre assistant repose sur la création d'interfaces graphiques sur les quelles, on communique avec l'ordinateur via la souris, en cliquant sur des boutons, des icônes, des cases à cocher ou avec le clavier pour remplir des formulaires.

Le principe repose sur l'introduction d'un langage intermédiaire entre un utilisateur et l'ordinateur. Ce langage aura la fonction d'intervenir sur un programme en rentrant des données d'un utilisateur et de sortir le résultat correspondant à ces données (figure n° 7)

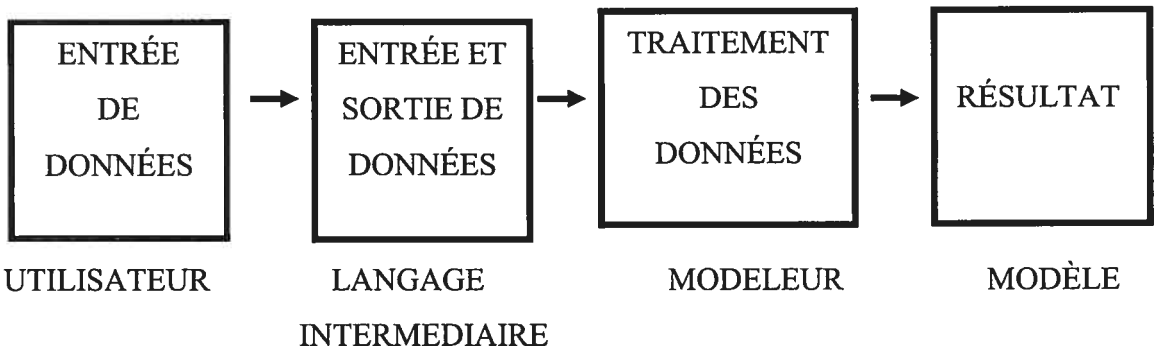


Schéma de fonctionnement de l'assistant

II.4.4.3.1. LE FICHIER D'ORIGINE :

Le fichier d'origine est un fichier existant correspondant à la description d'une famille d'objets. Ce fichier doit avoir une extension compréhensible par le logiciel 3D utilisé (Exemple : Txt). Selon l'objet à modéliser, Ce fichier doit être programmé de façon à ressortir les procédures permettant de générer un ensemble de modèles. Pour générer un modèle quelconque, on doit seulement intervenir sur les procédures de sa création.

II.4.4.3.2. LE FICHIER CRÉE :

Le fichier crée doit correspondre à la description du modèle recherché par le concepteur.
Afin de créer notre nouveau fichier, nous devons créer un programme qui aura pour tâche de :

- Lire le fichier d'origine en indiquant son répertoire, et le nom du fichier.
- Permettre d'apporter les changements nécessaires sur le fichier d'origine. Pour effectuer les nouvelles taches assignées, nous devons indiquer les lignes sur les quelles il faut apporter les changement en spécifiant les chaînes de caractère que nous voulons changer. Les variables sur lesquelles porteraient les changements peuvent être des valeurs numériques, des caractères ou même des chaînes de caractères.

- Créer le nouveau fichier en indiquant l'extension appropriée et le répertoire sur le quel il sera enregistré.

III) - CAS D'ÉTUDE : LES ESCALIERS.

Dans notre modélisation de l'escalier, compte tenu des spécificités techniques et conceptuelles auxquelles devra répondre ce dernier, nous avons voulu mettre en avant certains aspects. Comme la conception et la réalisation des escaliers dans la pratique font appel à la normalisation des éléments constitutifs, par souci de répondre à des effets d'esthétique, de confort et de faisabilité, nous avons jugé utile d'éviter certaines démarches qui donnent lieu à des situations et des résultats peu communs.

Ainsi, procéder par calculer le nombre et la dimension des marches de l'escalier en fonction de la hauteur de l'étage, peut donner des dimensions difficilement réalisables avec exactitude. Si on choisi certains exemples de hauteurs, voici les résultats (tableau n° 1) :

Hauteur de l'étage (en cm)	Nombre de marches	Hauteur de la contremarche (en cm)
283	15	18.8666666....
	16	17.6875
	17	16.647058....
303	16	18.9375
	17	17.8235...
	18	16.83333.....

Tableau n° 1 : calcul des contremarches en fonction de la hauteur de l'étage

Si on fixe la hauteur de l'étage comme paramètre de départ, certaines hauteurs peuvent donner des contremarches avec des dimensions irrégulières (portant plusieurs chiffres après la virgule).

Les solutions pratiques existantes pour pallier ce problème nous semblent peu satisfaisantes :

- Certains auteurs (W.Mannes, 1980) ont tendance à minimiser le problème en arrondissant la dimension de la contremarche (par exemple : 16.647058 à 16.65). Cependant, si le nombre de marche est important, le cumul de cette incertitude peut créer des malfaçons qui peuvent répercuter sur la qualité des travaux.

- Au cours de la réalisation, très souvent, on procède par des corrections sur la première marche. C'est à dire pour avoir la hauteur exacte entre les deux niveaux, on garde une même hauteur pour toutes les contremarches sauf pour la première où elle est généralement légèrement plus petite ou plus grande. Cette solution peut souvent nécessiter des efforts supplémentaires au cours de la réalisation.

Notre recherche n'a pas pour but de traiter toutes les démarches et les problèmes liés à la conception d'un escalier. Le choix de notre démarche est dicté par le souci d'éviter les incertitudes et les malfaçons pouvant découler d'une autre demarche.

III.1) : CHOIX DE L'ESCALIER:

Le choix de l'escalier comme application à notre méthode est motivé par plusieurs raisons :

- La création d'un modèle d'escaliers peut faire appel à plusieurs procédures successives, à savoir : les procédures de création d'une marche, de la première volée et puis de l'ensemble des volées qui peuvent constituer cet escalier.

- Malgré l'existence de plusieurs types d'escaliers que l'on peut classer selon leurs géométries (escalier droit à une ou plusieurs volées, hélicoïdal, balancé, pyramidal...etc.), si nous choisissons une façon intelligente de consigner les procédures de création de ces escaliers, il est possible de regrouper tous ces types d'escaliers dans un seul programme (un seul fichier). Cette description unique permettra d'éviter la redondance de l'information. Ainsi, par exemple au lieu de décrire pour chaque type

d'escaliers, sa propre procédure de création de sa marche (sa géométrie : giron, hauteur, sa forme générale), nous pouvons décrire une seule procédure de marche avec possibilité d'intervenir sur ses paramètres pour obtenir la marche recherchée.

- Un escalier peut être composé par une succession d'une infinité de types de volets. Grâce à la possibilité de pré programmer les intentions des acteurs sur le langage visuel. On peut construire notre escalier au fur et à mesure que le concepteur choisisse les caractéristiques de sa volée. C'est-à-dire grâce au langage visuel, on peut intervenir sur le fichier d'origine pour ajouter une succession de volée avec les caractéristiques souhaitées par le concepteur. Cette méthode permet de récupérer en plus du modèle, la description de l'objet crée.

III.2. CHOIX DES LANGAGES ET LOGICIELS DE PROGRAMMATION :

III.2.1. INTRODUCTION :

La méthode que nous proposons n'est pas dépendante d'un logiciel ou d'un langage spécifique. Son avantage réside justement sur la possibilité de l'appliquer à tous les langages et logiciels. Aujourd'hui, il existe beaucoup de langages et des logiciels au point qu'il est difficile de faire un choix. Afin de vérifier nos hypothèses, nous avons opté pour des logiciels et des langages en conformités avec nos objectifs et qui répondent amplement à la direction que nous avons donnée à notre recherche.

III.2.2) : CHOIX DE POV-RAY POUR LA DESCRIPTION DES PROCEDURES DES ESCALIERS :

Si la conception architecturale fait appel à plusieurs acteurs qui peuvent communiquer sur tous les phases de réalisation d'un projet architectural. Ces communications peuvent concerner l'aspect formel, esthétique, structurel...etc. Le choix du logiciel Pov-ray est dicté par la possibilité de l'utiliser par tous les acteurs de la conception à toutes les phases de la conception

architecturale. Ce logiciel assure le passage d'une phase en amont à une phase en aval sans pour autant changer de logiciel. Le résultat obtenu lors d'une phase en amont peut être directement utilisé dans une phase en aval.

III.2.3 : CHOIX DE JAVA :

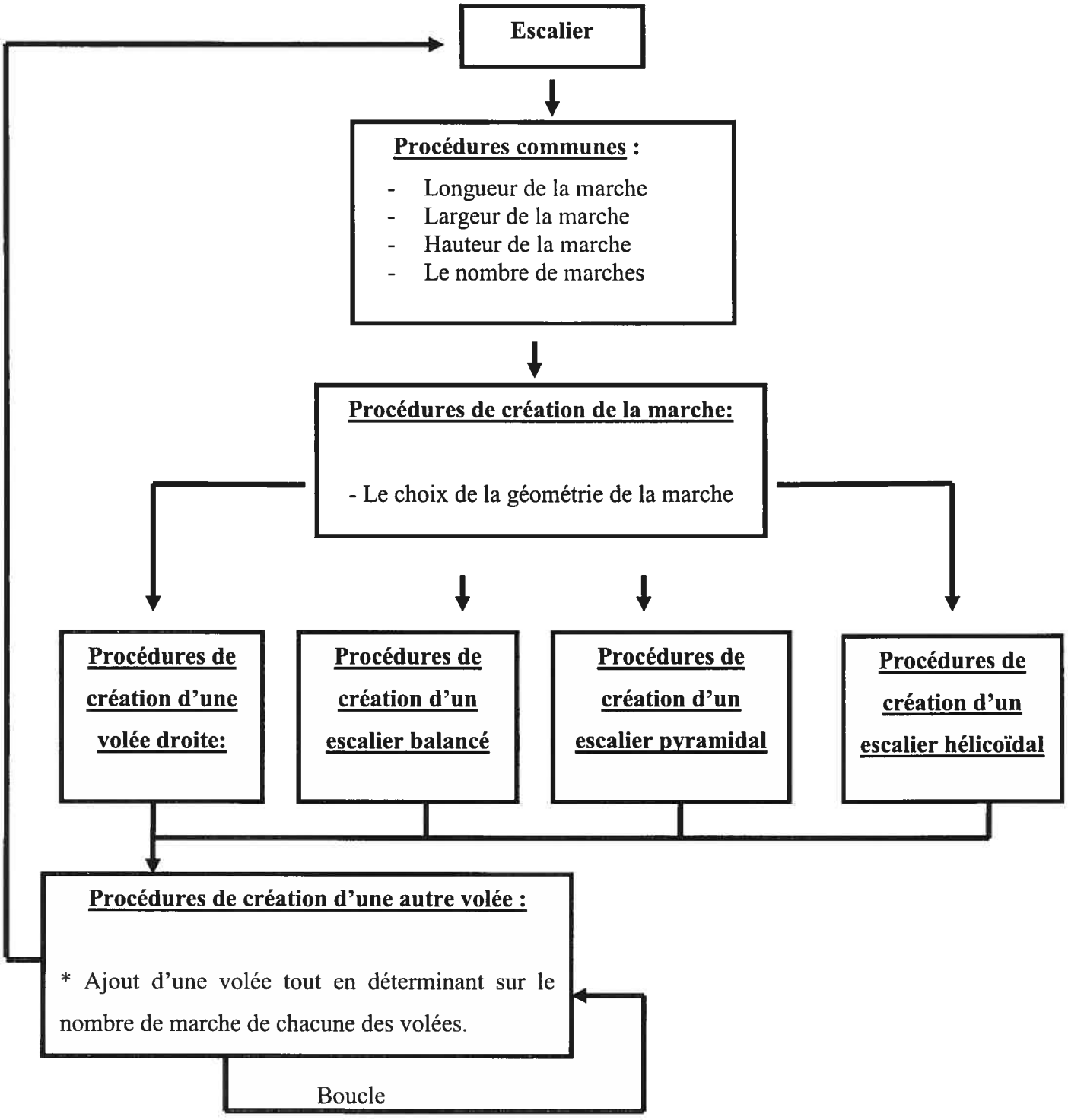
Le but de notre recherche n'est pas de comparer la perfection des logiciels ou des langages, vu que les nouvelles technologies sont en mutations quotidiennes. Afin que notre assistant touche le plus grand nombre de personne, nous avons opté pour java. Java est un langage de programmation objet qui fut inventé par Sun Microsystems en 1995. Il permet de manipuler des objets, éléments définis par des classes. Chaque classe permet d'attribuer un type donné à des objets. Ces classes définissent entre autres, des membres (données qui constituent un objet), ainsi que des méthodes (actions réalisables par un objet).

L'un des points forts de ce langage est la possibilité de lecture de ses applications sur tous les systèmes d'exploitation existants. Ses applications fonctionnent à la fois sur tous les systèmes d'exploitation Linus, Windows, Macintosh ou autre. Tout programme écrit en Java est exécuté dans tout système d'exploitation. Java est doté d'une riche bibliothèque de classes couvrant tous les domaines existants. Il offre la possibilité de créer toute sorte de projets de façon efficace et rapide.

III.3. PROCEDURES DE CRÉATION D'UN ESCALIER :

Afin de construire un modèle 3D, tout en évitant les problèmes de redondance, d'incohérence, d'inconsistance ou de mise à jour de l'information, nous avons opté pour une stratégie permettant de codifier l'escalier d'une façon hiérarchisée. On codifie du plus commun des types d'escaliers au plus particulier. Cette façon nous permettra d'écrire un seul programme pour tous les escaliers. Pour passer d'un type d'escalier à un autre un concepteur n'a qu'à intervenir sur les paramètres de l'escalier en question.

III.3.1. SCHEMA DE PROCEDURES DE CRÉATION D'UN ESCALIER :

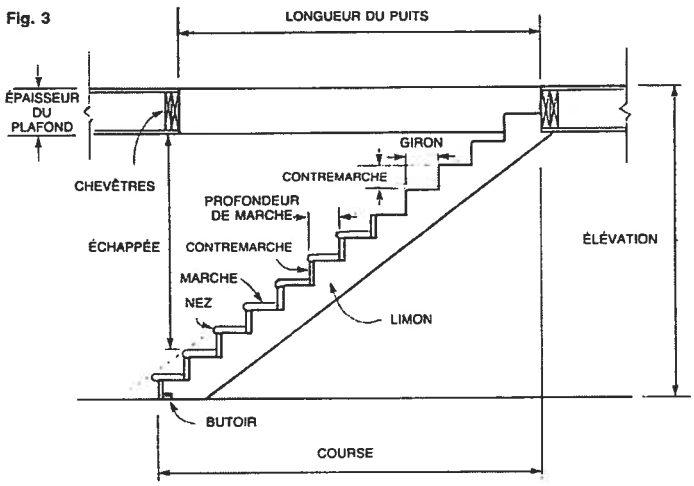


III.3.2) : PROCEDURES COMMUNES POUR LA CRÉATION DE TOUT ESCALIER :

Un escalier commode et normalement conçu, satisfait à la relation: "g + 2h = 0,64m".

Formule de Blondel **2h+g = 60 à 65.**

- "h" : est la hauteur de la marche qui peut varier de 0.16 à 0.18 m, selon ce que l'on veut un escalier plus ou moins doux.
- "g" : le giron.
- "H" est la hauteur entre planchers.
- "n" est le nombre de marche : n = H/h. Il est nécessaire que "n" soit un nombre entier.



CONTREMARCHE : Cloison fermant l'ouverture entre deux marches et sur laquelle vient s'appuyer la marche. Se dit aussi de la partie verticale du limon. Se dit aussi de la hauteur de cette partie verticale (dans ce cas on emploie aussi les termes « hauteur de marche » ou « montée »). On utilisera souvent les initiales « CM ».

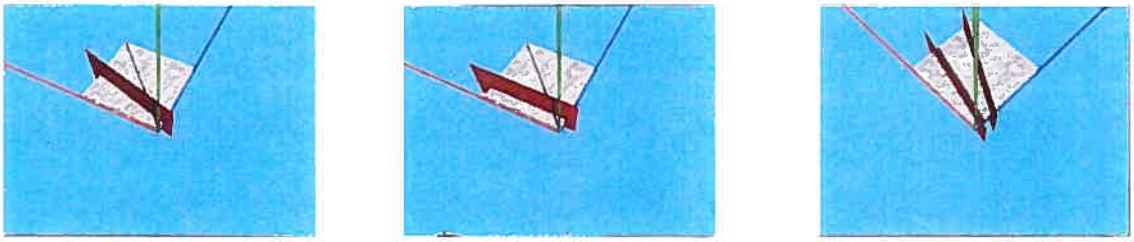
Source : Les publications du Québec (1985).

Figure n° 06 : Terminologie de l'escalier.

III.3.3. PROCEDURE DE CRÉATION DE LA MARCHE:

Afin d'éviter la réécriture de la procédure de création de la marche pour chaque type escalier, il est possible de créer une seule procédure qui permet de regrouper toutes les marches. Cette procédure consiste à créer un parallélépipède que l'on coupe avec deux plans verticaux.

Pour choisir la marche souhaitée, il suffit d'intervenir sur les paramètres géométriques de la marche (largeur, et hauteur) et tout en tronquant le volume avec un ou deux plans verticaux (figure n° 09)



Hélicoïdal droit balancé

Figure n° 07 : Procédures de création de la marche.

On peut passer d'une marche d'un escalier droit à un escalier balancé, hélicoïdal ou autre sans pour autant écrire pour chaque type d'escalier, une procédure de création de sa marche.

Noter que dans tous les cas de types d'escaliers, la marche est la même alors que dans le cas de l'escalier balancé, il y a autant de nombre de marche dans une volée d'escalier, qu'il y a de forme de marche. C'est pourquoi, il faut programmer de façon à avoir pour chaque marche sa propre forme.

III.3.4 PROCEDURE DE CRÉATION DE LA VOLÉE :

En optant pour un type de marche particulier, l'usager aurait déjà choisi le type de la première volée de l'escalier. A ce niveau, chaque type de volée présente sa propre procédure de création.

Pour chaque type de volée, l'usager doit intervenir sur les propres paramètres de cet escalier. Ainsi, par exemple :

- Si on choisit l'escalier droit, nous avons juste à intervenir sur le nombre de marche et sur la possibilité d'ajouter un palier de repos ou pas (figureS n° 08 & n°09).

Exemple :

```
#declare volée1=
 #déclare n1= nombre de marche à fixer;
#déclare a1 = 90;  // angle de rotation de la volée
                #declare Count = 0   ;
                   #while (Count  < n1  )
                   object { marche translate < g*Count , h*Count ,0> rotate 0*y }
                    #declare Count = 1+  Count ;
                    #end
    object  { volée1 Rotate_Around_Trans( y*a1,< 0 ,0 ,0>  ) }
```

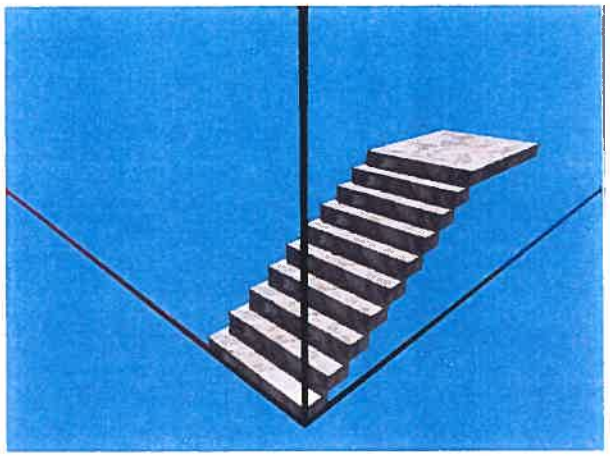


Figure n° 08 : Escalier droit à une seule volée avec palier de repos.

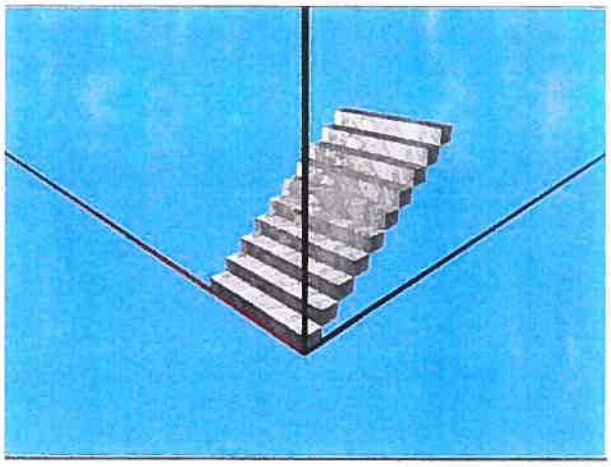


Figure n° 09 : Escalier droit à une seule volée sans palier de repos.

- Dans le cas d'un escalier hélicoïdal, nous devons choisir en plus du nombre de marche, le rayon du poteau central. Ce rayon permet de déterminer la largeur de la marche au collet. Nous pouvons choisir entre un escalier hélicoïdal avec ou sans poteau central (figures n° 10& n°11).

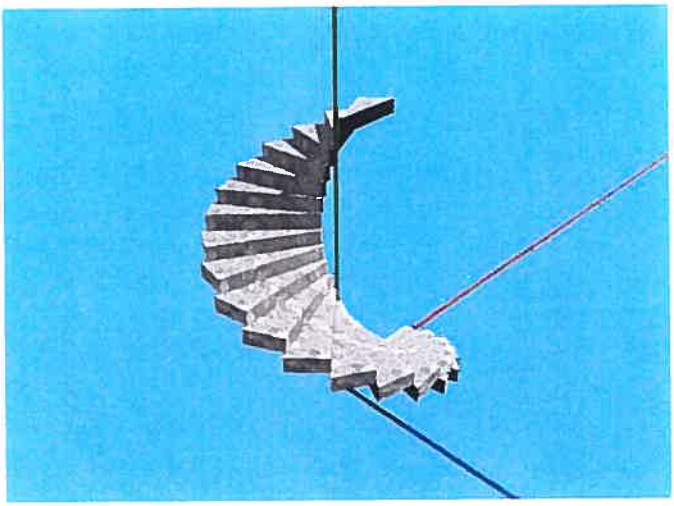


Figure n° 10 : Escalier hélicoïdal sans poteau central.

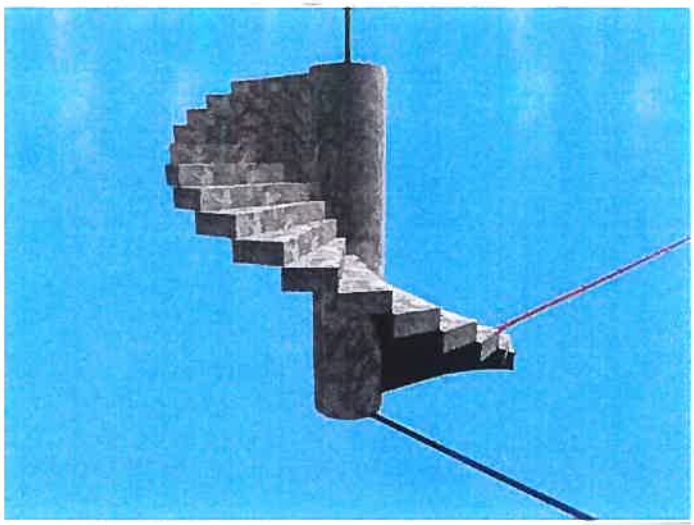


Figure n° 11 : Escalier hélicoïdal avec poteau central.

- L'escalier balancé doit répondre à certaines exigences particulières en plus de celles qui consistent à fixer le nombre de marche. Il existe plusieurs façons de calculer un escalier balancé. Nous avons opté pour la méthode de calcul, où le principe consiste à fixer la dimension de la dernière marche du collet. (Le collet désigne la plus faible largeur d'une marche dans un escalier balancé).

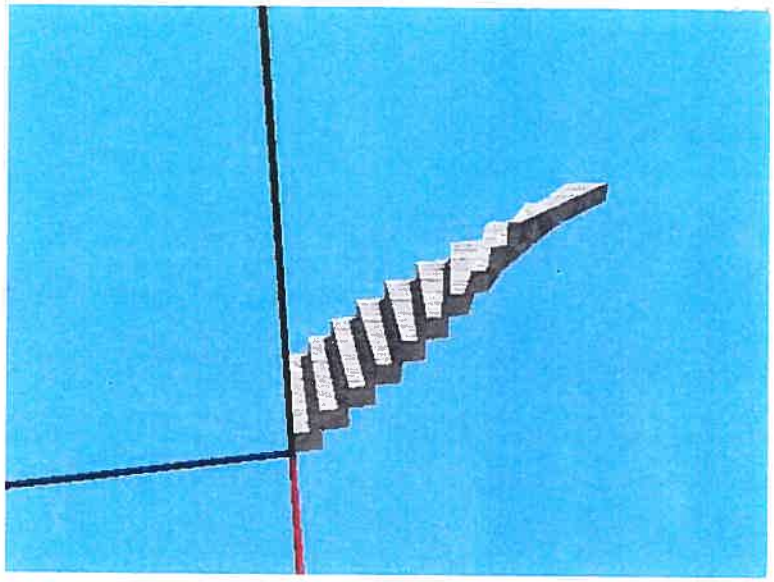


Figure n° 12 : Escalier balancé.

Cette dimension devrait être inférieure à celle du giron (dimension d'une marche normale). Ainsi, par exemple si nous avons à concevoir un escalier à 5 marches, nous devons calculer la différence de la longueur du collet et le nombre de marches par la dimension de la derrière marche que l'on a fixé, c'est-à-dire 5 fois cette dimension. Cette différence nous permettra de calculer la part à ajouter à chaque marche.

Exemple : Marche 1 -----1 part

Marche 2 ----- 2 parts

Marche n ----- n parts

Une part sera égale au rapport entre la différence de la longueur du collet et n fois la dimension de la dernière marche que l'on divise par la somme des nombres de parts.

- Dans le cas d'un escalier pyramidal, la procédure de création de cet escalier consiste à translater la marche de départ tout en réduisant ses dimensions (figure n°13).
-

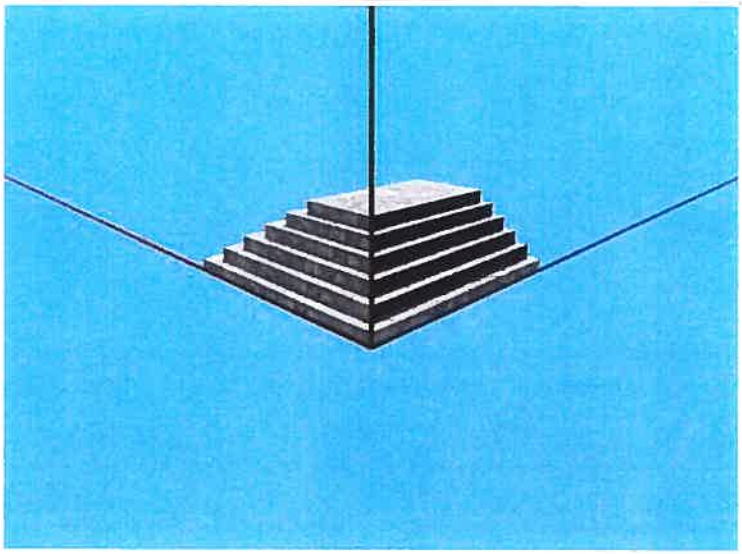


Figure n° 13 : Escalier pyramidal.

III.3.5. PROCÉDURES DE CRÉATION D'UNE SUITE DE VOLÉES: LE CAS D'UN ESCALIER DROIT AU NOMBRE DE VOLEÉS INDETERMINÉ :

III.3.5.1 CRÉATION DE LA DEUXIEME VOLÉE :

Le programme du fichier d'origine existant ne contient que les procédures de génération de la première volée et celles de la deuxième volée. Pour obtenir la deuxième volée, nous avons recopié le même code de la première volée tout en donnant le choix à un utilisateur d'intervenir sur le nombre de marche, sur son palier de repos et sur l'angle de rotation de la deuxième volée par rapport a la première volée.

```
#declare volee1 =          // pour ajouter une volée
    #declare vol =  1; // donner le choix permettant d'ajouter la deuxième volée
  #if( vol = 1)
        #declare a1 = 0;  // angle // le choix de l'angle de rotation de la deuxième volée par a
                              la première
```

Pour créer la deuxième volée avec le nombre de marche souhaitée, nous avons recopié le code de la première volée tout en paramétrant le nombre de marche.

```
#declare marche1 =

        #declare n1 = 6    ; // nombre de marche de la deuxième volée

        #declare Count = 0    ;

        #while (Count  < n1  )

        object { marche translate < g*Count , h*Count ,0> rotate 0*y }

         #declare Count = 1+  Count ;

         #end

 object  { marche1 Rotate_Around_Trans( y*a1,< 0 ,0 ,0>  ) }
```

Pour déplacer la volée ainsi crée, nous avons codifié les différents cas de rotation que peut avoir la deuxième volée par rapport à la première (0 -90 ,90 ou 180 degrés). En fonction de ce choix, nous devons choisir la translation à effectuer.

```
            #if ( a1 = 0)  // ajouter une volée droite directe

            object {volee1 translate  < g*(n-1)+m , h*n ,0> }

            #end


            # if( a1 = 90) // ajouter une volée droite tout en tournant 90 degré

            object {volee1 translate  < g*(n-1) , h*n ,0> }

            #end


            #if( a1 = -90) // ajouter une volée droite tout en tournant -90 degré

            object {volee1 translate  < g*(n-1)+m , h*n ,m> }

            #end


            #if (a1 = 180)    // ajouter une volée droite tout en tournant 180 degré

            object {volee1 translate  < g*(n-1) , h*n ,m1> }

            #end

#end
```

III.3.5.2. PROCEDURES DE CRÉATIONS D'AUTRES VOLÉES :

Pour créer la troisième volée, la codification devient de plus en plus difficile pour enregistrer tous les cas possibles, c'est pour quoi que nous avons profité de la puissance du langage java qui permet d'intervenir sur un fichier et d'apporter les changements souhaités tout en créant un nouveau fichier avec les nouvelles données.

Ainsi, les procédures de rajouts de volées à l'escalier seront crées au fur et à mesure que le concepteur décide d'un choix de caractéristiques de la volée à ajouter. Le langage visuel de l'assistant se chargera de copier le code de la première volée et de le réécrire tout en demandant au concepteur de choisir les caractéristiques qu'il souhaite donner à sa volée. Ainsi, le concepteur devra choisir le nombre de marche et l'angle de rotation de la volée ajoutée par rapport à la précédente (0, 90,-90 ou 180 degrés).

Selon l'angle et le nombre de marche choisis, l'assistant se chargera d'ajouter des valeurs prédéfinies pour déplacer la volée à la place souhaitée. Les valeurs prédéfinies des translations possibles sont jointes sur le tableau n°.

Pour créer la troisième volée, il faut d'un part recopier le code de la première volée tout en apportant les caractéristiques souhaitées et d'autre part savoir l'angle de la rotation de la volée précédente (la deuxième volée).

Exemple

Si l'angle de rotation de la deuxième volée (a1 = 0) (figure n° 14)

```
#if ( a1 = 0) // ajouter une volée droite directe
        object {volee1 translate < g*(n-1)+m , h*n ,0> }        #end
```

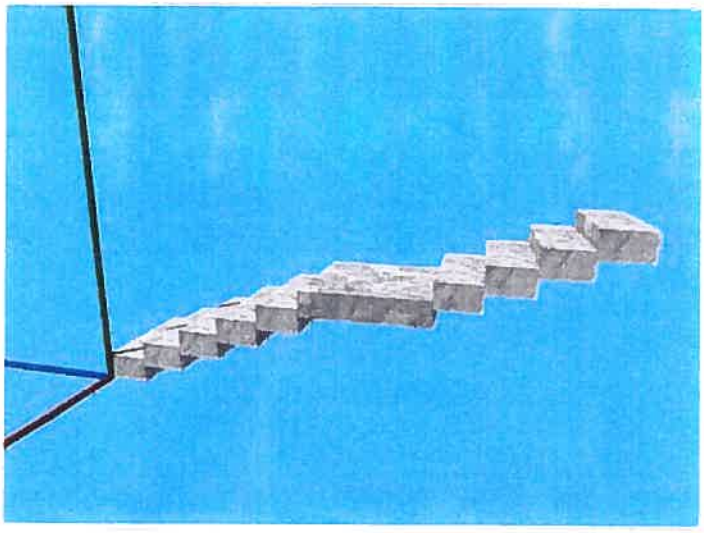


Figure n° 14 : Escalier droit à deux volées.

Pour déplacer la troisième volée le concepteur devra choisir l'angle de rotation

Exemple

Si l'angle de rotation de la troisième volée (a1 = 0) (figure n° 15)

```
#if (a1 = 0)  // ajouter une volée droite directe

        object {volee2 translate  < g*(n-1) +m +g*(n1-1) +m, h*(n+n1), 0 > }
#end
```

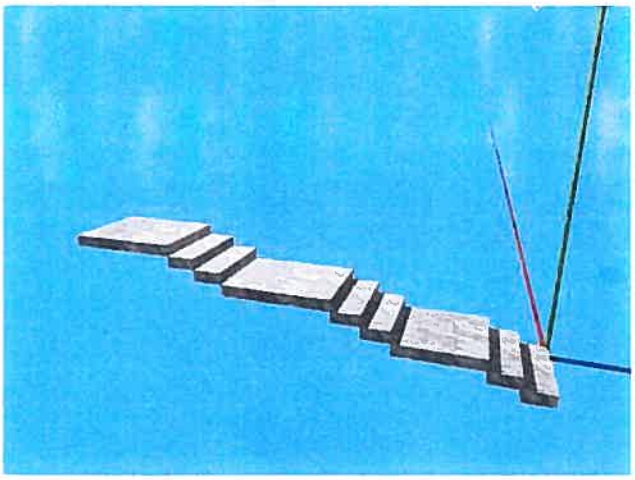


Figure n° 15 : Escalier droit à trois volées.

Ainsi à chaque fois que l'angle précèdent (a1 = 0) et l'angle choisi pour la nouvelle volée est aussi (a1 = 0) l'assistant se chargera juste d'ajouter :

- g*(nv-1) +m : à l'abscisse x // nv est nombre de marche de la volée
- h* nv : à l'ordonné y
- 0 : à la cote z

Le tableau ci-dessous montre les valeurs à ajouter pour translater la nouvelle volée crée.

III.3.5.3) : VALEURS À AJOUTER POUR OBTENIR N'IMPORTE ESCALIER DROIT :

Angle Précèdent	Angle suivant	Ajout	Observation
Angle a = 0	Angle a = 0	g*(n-1) +m +**g*(n1-1) +m**, h*(n+**n1**), 0	Ajouter +g*(n1-1) +m à X
	Angle a = 90	g*(n-1) +m +**g*(n1-1)**, h*(n+**n1**), 0	Ne pas ajouter de m à X
	Angle a = -90	g*(n-1)+m +**g*(n1-1)+m** , h*(n+**n1**) , m	Même chose que a = 0 tout en ajoutant m à Z
Angle a = 90	Angle a = 0	g*(n-1) +**m**, h*(n+**n1**), -g*(n1-1)-m	Ajouter m a X et -g*(n1-1)-m à Z
	Angle a = 90	g*(n-1), h*(n+**n1**) , -g*(n1-1)-m	-g*(n1-1)-m à Z
	Angle a = 180	g*(n-1), h*(n+**n1**), -g*(n1-1)	-g*(n1-1) à Z
Angle a = -90	Angle a = 0	g*(n-1) +m , h*(n+**n1**), m + **g*(n1-1)**	Ajouter +g*(n1-1) à Z
	Angle a = -90	g*(n-1)+m +**g*(n1-1**,h*(n+**n1**),m+ **g*(n1-1)+m**	Ajouter +g*(n1-1) +m à Z
	Angle a = 180	g*(n-1)+m-**m** , h*(n+**n1**) , **m** + **g*(n1-1)**	Ôter – m à X Ajouter +g*(n1-1) +m à Z

angle a = 180	Angle a =9 0	-g*(n1-1)-**m**, h*(n+**n1**), -**m**	Ôter –g*(n1-1)-m à X Et – m à Z
	Angle a = -90	-g*(n1-1), h*(n+**n1**), 0	Ôter –g*(n1-1)-m à X
	Angle a = 180	-g*(n-1)-m, h*(n+**n1**), 0	Ôter -g*(n-1)-m à X

Tableau n° 02 : valeurs à ajouter pour obtenir n'importe quel escalier droit.

III.3.5.4) : QUELQUES RESULTATS :

- RESULTAT N° 1

Ce résultat est obtenu suite à une rotation de la volée de 90 degrés (figure n°16).

```
# if (a1 = 90) // ajouter une volée droite tout en tournant 90 degré
object {volee1 translate < g*(n-1) , h*n ,0> }
#end
```

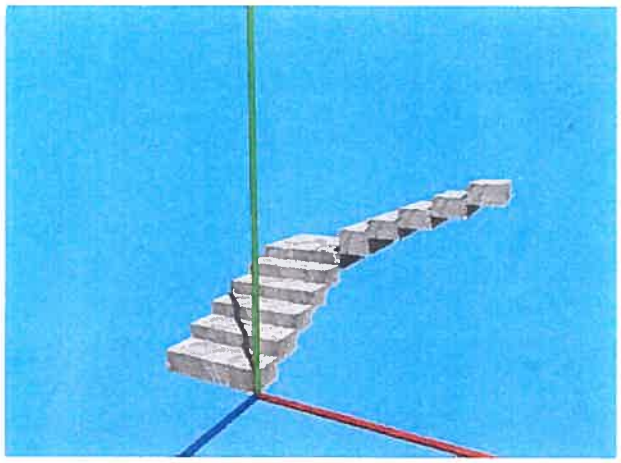


Figure n° 16 : Escalier droit avec la deuxième volée tournant à 90 degrés.

- RESULTAT N° 2

Pour obtenir ce résultat, nous devons translater la deuxième volée de 90 degrés et en suite la troisième volée de 180 degrés (figure n° 17)

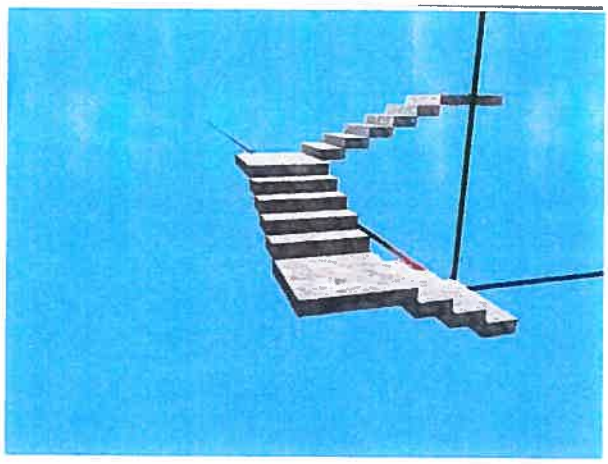


Figure n° 17 : Escalier droit avec la deuxième volée tournant à 90 degrés et la troisième tournant à 180 degrés

- RESULTAT N° 3

Ce résultat est obtenu suite à une rotation de la deuxième volée de 180 degrés (figure n° 18)

```
#if (a1 = 180)    // ajouter une volée droite tout en tournant 180 degré
        object {volee1 translate < g*(n-1) , h*n ,m1> }
        #end
```

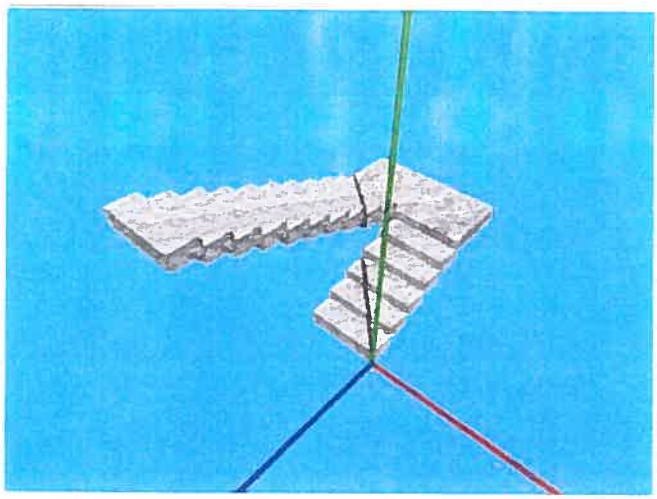


Figure n° 18 : Escalier droit avec la deuxième volée tournant à 180 degrés.

III.4. ASPECT INTERACTIF DE L'ASSISTANT :

III.4.1. PRINCIPE DE FONCTIONNEMENT :

Afin de générer un modèle, un concepteur doit intervenir sur le fichier d'origine et apporter les changements nécessaires. Notre démarche de travail repose sur deux types d'actions que l'on peut opérés sur nos programmes d'origines :

III.4.1.1 : CHANGEMENT DE PARAMÉTRES :

Grâce à l'assistant, on peut intervenir sur les paramètres de l'objet en opérant des changements sur des valeurs numériques, des caractères ou des chaînes de caractères. Chaque changement opéré permettra de générer un modèle unique aux caractéristiques désirées par le concepteur. Pour réaliser cette opération, le principe, c'est d'agir sur la description d'un programme (méthode : LireCreer) qui consiste à lire le fichier d'origine, ligne par ligne et colonne par

colonne, tout en lui attribuant les instructions d'apporter des changement à des adresses mémoires précises (numéro de la ligne et numéro de la colonne).

Ainsi, par exemple, pour changer la hauteur de la marche de l'escalier, il suffit de programmer de façon que la méthode LireCreer apporte des changements à la ligne et la colonne correspondant à la valeur de la hauteur de la marche de l'escalier.

Une fois que tous les changements sont opérés, l'assistant se chargera de créer et d'ouvrir un nouveau fichier en un format compréhensible par le logiciel. Un exemple de procédure de fonctionnement en langage java est disponible en annexe.

III.4.2 : AJOUT OU SOUSTRACTION AU PROGRAMME :

On peut ajouter ou soustraire une partie de programme au fichier d'origine. En effet, dans le cas de l'escalier droit, le fichier d'origine ne comporte que la description d'une seule volée. Pour créer un escalier à volées multiples où chaque volée pourrait avoir ses propres caractéristiques particulières (nombre de marches, position par rapport à la voilée précédente...etc.), le concepteur n'a pas à écrire le code relatif aux volées ajoutées, l'assistant se chargera d'ajouter le programme correspondant au fur et à mesure que le concepteur choisisse les caractéristiques des volées.

Comme dans le cas de changement de paramètres, le principe se base sur la création d'une méthode LireCréer qui consiste à lire le fichier d'origine ligne par ligne et colonne par colonne et copier les chaînes de caractères existants entre deux lignes correspondantes au programme que nous désirons réécrire tout en donnant la possibilité au concepteur d'intervenir sur les paramètres de ce programme.

Ainsi, pour créer un escalier à volées multiples, l'assistant se chargera d'écrire son programme en procédant par la réécriture de la volée de départ tout en apportant les changements souhaités par le concepteur.

III.4.2. SUITE DE FENÊTRES DE DIALOGUE :

III.4.2.1. PREMIERE FENÊTRE :

Afin de rester dans la hiérarchisation des procédures, nous avons conçu la première boite de dialogue de façon à permettre au concepteur de choisir visuellement le type d'escaliers. Le langage visuel se chargera d'intervenir sur les caractéristiques géométriques de la marche (dimensions et forme) en apportant les modifications sur les plans verticaux permettant de varier la forme de la marche (figure n° 19).

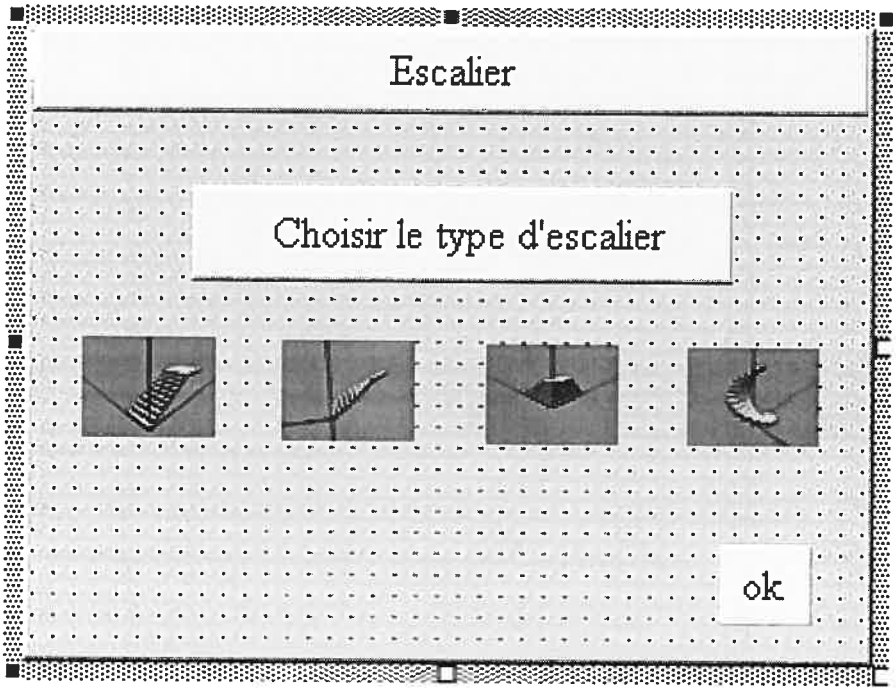


Figure n° 19 : Première fenêtre de dialogue.

III.4.2.1.2. DEUXIEME FENÊTRE :

Si nous choisissons type d'escalier, une deuxième fenêtre devrait s'ouvrir

Pendant que l'usager choisit les types d'escaliers, le langage java se chargera grâce à ses classes (LireCreer, Substring…) d'écrire le code du fichier en question.

Ainsi pour ajouter un autre type d'escalier droit, java se chargera de copier une partie de code de l'escalier droit existant par défaut, tout en nous proposant d'apporter les modifications nécessaires sur ce fichier.

Sachant qu'un escalier peut être composé par un ensemble indéterminé de volées, nous avons pensé à créer une boucle de fenêtres de dialogues qui s'arrêtera une fois que l'usager ne voudrait plus ajouter de nouvelle volée (figure n° 20).

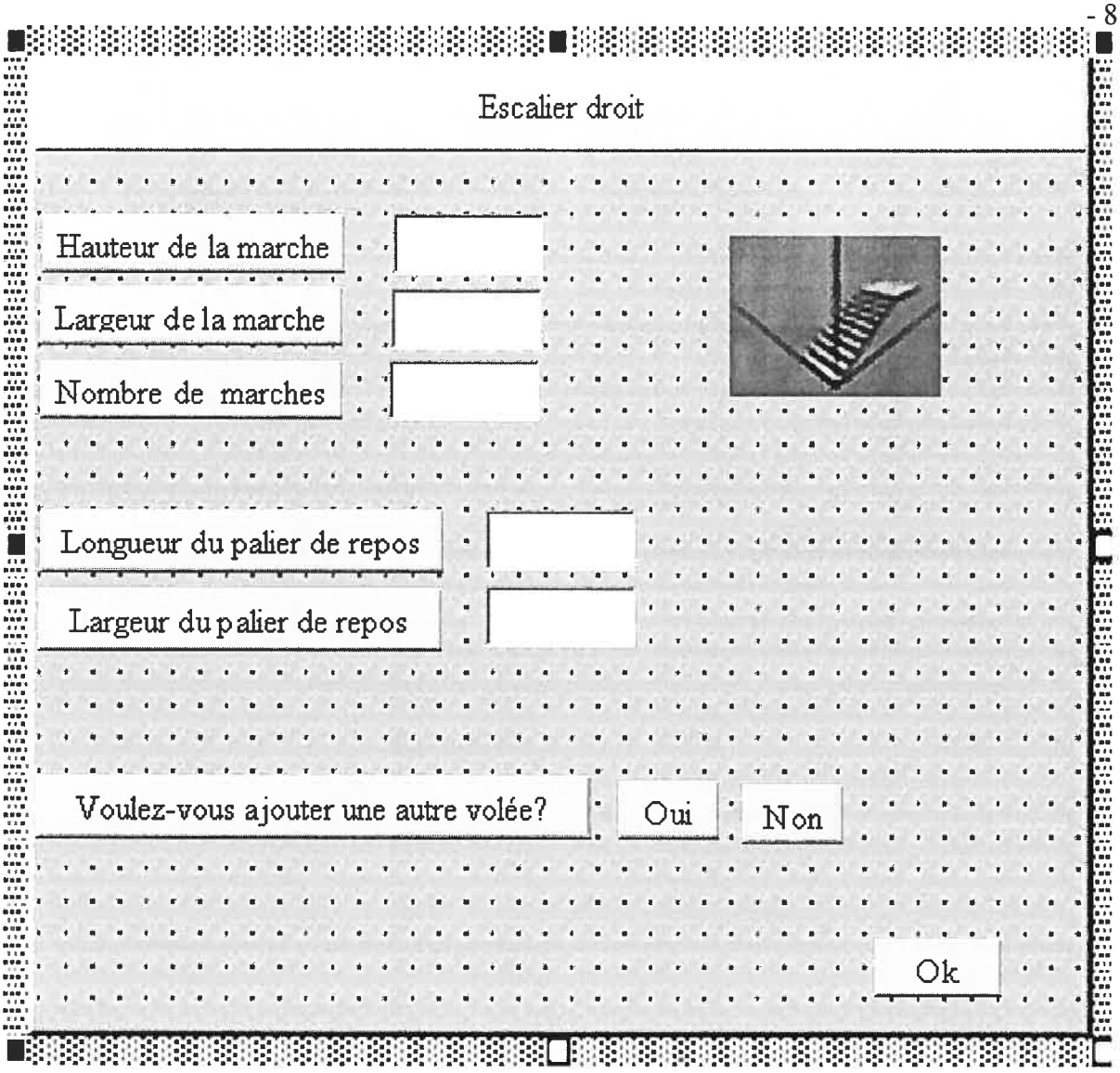


Figure n° 20 : Deuxième fenêtre de dialogue.

III.4.3. RELATION ENTRE LE JAVA ET LE MODELEUR 3D:

Une fois le nombre de volée arrêté , le langage java se chargera en premier lieu de créer le nouveau fichier avec toutes les nouvelles données et ensuite, il lancera le logiciel Pov ray avec le nouveau fichier ainsi crée .

Grâce à certaines méthodes de java, il est possible de lancer un logiciel si nous indiquons clairement le répertoire de son fichier exécutable. Ce logiciel sera lancé directement de l'interface, tout en générant le modèle souhaité.

Selon l'encyclopédie informatique libre (CCM) : comment ça marche (www.commentcamarche.net), dans le langage java, on appelle méthode une fonction ou un sous programme qui peut exécuter dans plusieurs parties de programme, un ensemble d'instructions par simple appel de la fonction dans le corps du programme principal. Cela permet de simplifier le code pour le réduire à la taille minimale. Avant de l'utiliser, une méthode doit être définie pour permettre au compilateur de la reconnaître, c'est-à-dire de reconnaître son nom, ses arguments et les instructions qu'elle contient (http://www.commentcamarche.net/java/javafonc.php3).

En effet, Java peut exécuter une commande système, c'est-à-dire, il peut transmette au système d'exploitation une commande destinée normalement à une console (fenêtre MS-DOS sous Windows). Ce genre de fonctions sous Java permet entre autres, de lancer des logiciels externes.

Afin d'exécuter cette procédure, nous devons créer un objet **String** qui contient les commandes exactes que l'on veut écrire sous la ligne de commande, par exemple:

1 - Créer un objet **String** qui contient la commande exacte.

Par exemple : Sous Windows : Le cas du Pov-ray

```
String[] command = new String[3];
command[0] = "C:\\PROGRAM FILES\\POV-RA- 3.6\\bin\\pvengine.exe ";
command[1] = "C:fichierCree.pov";
command[2] = "sortie.bmp";
```

Selon le type de logiciel à lancer, nous devrions créer une suite de commande ou chaque commande devrait correspondre à une action à exercer. Dans certains cas de logiciels de modélisation, le fichier crée diffère du fichier récupéré après le lancement du logiciel, c'est pourquoi qu'il faut toujours ajouter une commande et le nom du fichier de sortie.

2 - Créer un objet "Runtime" et lui assigner le "runtime" (c'est à dire l'environnement d'exécution, le noyau du système en quelque sorte) courant :

```
Runtime runtime = Runtime.getRuntime ();
Process process = null;
Process proc = runtime.exec (command); // lance l'exécution.
```

III.5. APPLICATION DES ESCALIERS AU PROJET DU GLASS HOUSE :

III.5.1. INTRODUCTION :

Afin de rendre compte de l'utilité de notre assistant, nous avons jugé utile de voir comment, l'outil informatique peut-il aider un acteur de la construction pour placer l'escalier généré dans un projet quelconque sans pour autant être obligé d'écrire le programme de cet escalier sur le fichier du projet en question.

Pour cela, nous avons opté pour le projet du glass house qui est un pavillon d'exposition fait par l'architecte Bruno Taut en 1919 à Cologne en Allemagne. Ce choix est motivé par l'existence au sein de cet ouvrage de plusieurs types d'escaliers (droit, balancé, hélicoïdal, pyramidal).

III.5.2. PLACEMENT DES ESCALIERS DANS LE PROJET DU GLASS HOUSE :

Afin d'appliquer un escalier à un projet quelconque, deux opérations sont nécessaires :

- Choisir l'emplacement de l'escalier en déplaçant l'escalier. Pour cela, nous avons créé une procédure qui permet de déplacer l'escalier à un point quelconque.

```
object {choixdelescalier translate < -7, 2*h, -2.1 > rotate 180*y}
```

- Ajouter sur le fichier du glasshouse.pov, # include "nom du fichier de l'escalier.inc"

Grâce au langage visuel, on peut écrire un programme (une méthode) que l'on peut appeler LireCréer1 qui permet de lire le code du projet en question et d'ajouter une ligne de code : #include "nom du fichier .inc"// et qui doit avoir une extension (.inc) au lieu de (.pov)

On peut ajouter sur le fichier du glass-house.pov, autant de lignes qu'il y aurait d'escaliers. Afin de placer ces huit escaliers, nous avons juste à ajouter sur le fichier glass-house.pov :

```
#include "esc-pyramide.inc"   // escalier pyramidal

#include "escalier-principal.inc"  // escalier principal

#include "esccolimacon.inc"

#include "balance-d1.inc"

#include "bal-gauche.inc"

#include "colimac.inc"

#include "escArDroit2.inc"

#include "escArGauche1.inc"
```

III.5.3.ILLUSTRATIONS :

Nous avons présenté une illustration de notre démarche opératoire sur les figures suivantes (figures n° 21, n° 22 et n°23) où chacune d'elle représente une séquence de notre intention d'adaptation de nos escaliers modèles au contexte du glass house

En effet la figure n°21 illustre comment ajouter trois (03) types d'escaliers au projet du glass house :

- Un escalier pyramidal
- Un escalier droit
- Un escalier hélicoïdal

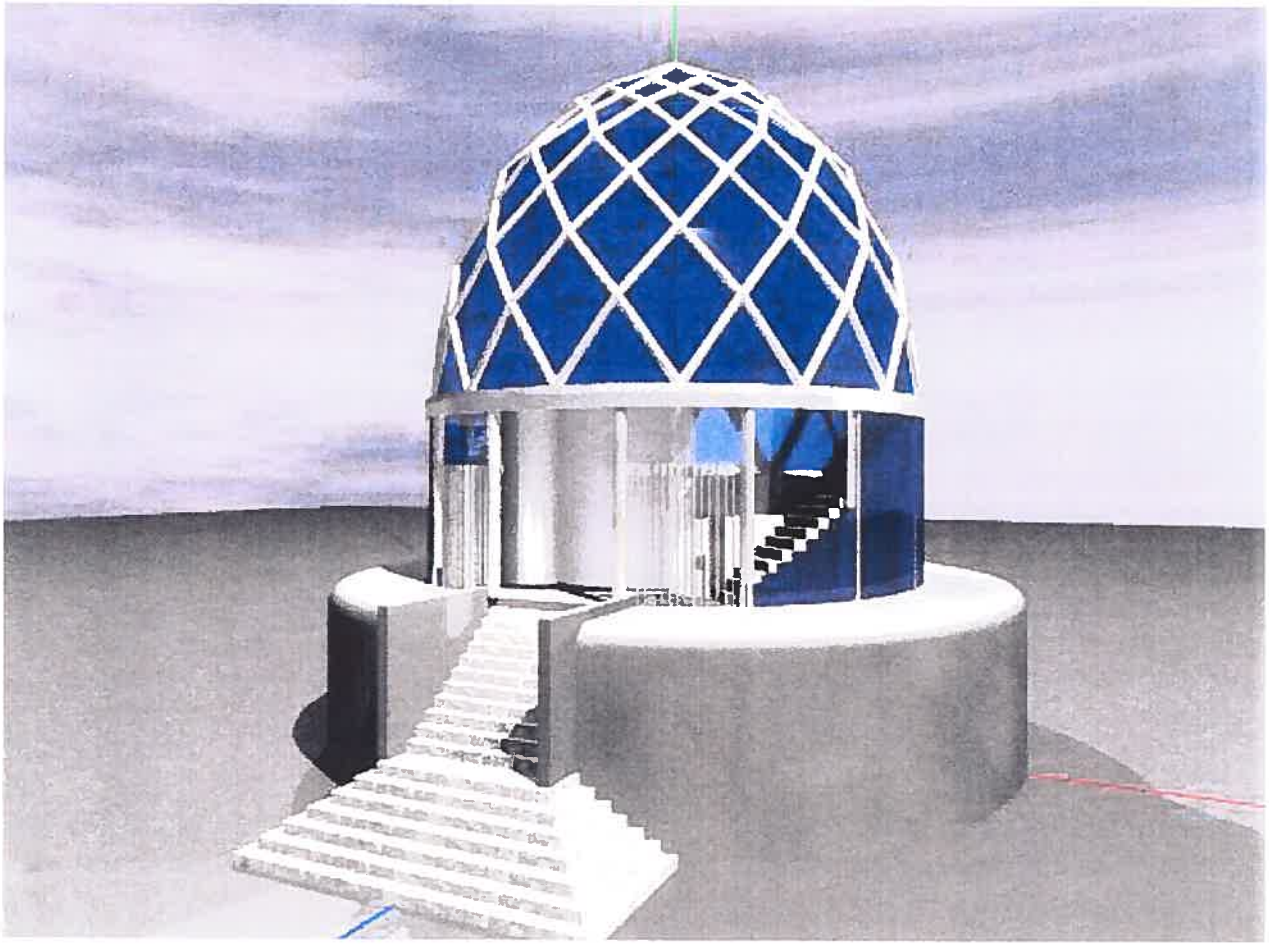


Figure n° 21 : Entrée principale du glass house (placement de 3 escaliers).

La deuxième illustration montre la rentrée arrière du glass house où nous avons appel a deux modèles d'escaliers (figure n° 22 et n° 23).

- Un escalier composé de deux volées droites.
- Un escalier balancé.

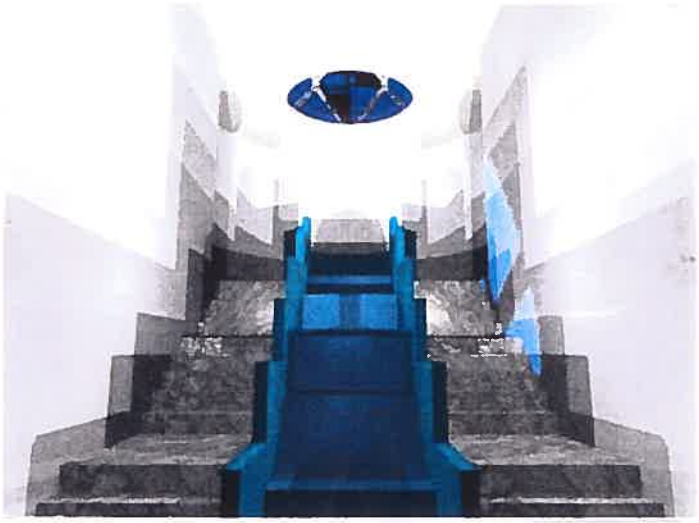


Figure n° 22 : Entrée arrière du glass house (placement de 2 escaliers).

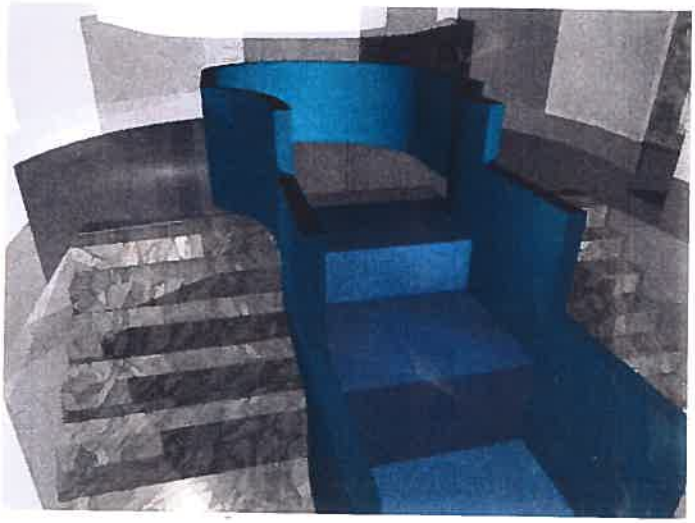


Figure n° 23 : Arrivée sur le hall central du glass house (placement de l'escalier balancé).

III.6) : CONCLUSION DU CHAPITRE :

Les résultats obtenus au terme de notre recherche nous montrent qu'il est possible qu'un outil informatique puisse servir un acteur de la conception architecturale dans le cadre de ses compétences informatiques.

Par le bais de notre cas d'étude, nous venons de démontrer que si nous associons les deux approches procédurales et visuelles, on peut, non seulement générer un modèle d'un escalier et le placer sur un projet architectural quelconque sans avoir une grande compétence dans le domaine de la programmation, mais on peut aussi récupérer la description de ce modèle. Ce qui peut être d'un apport très appréciable pour les phases suivantes.

En effet, pour générer, un escalier droit aux volées multiples, notre méthode se base sur la possibilité de coder les intentions des concepteurs. Le recours aux deux langages choisis pour notre méthode, nous permet de récupérer le programme de l'escalier sans avoir aucune connaissance en programmation.

Ainsi, par notre méthode, un acteur de la conception architecturale peut utiliser tout logiciel à base de programmation pour générer un modèle d'objet et récupérer son programme.

IV. CONCLUSION GÉNÉRALE:

IV.1 DISCUSSION DES RESULTATS :

Les outils actuels d'aide à la conception présentent certaines discordances par rapport aux besoins des concepteurs.

Au termes de notre recherche, le constat sur la disproportion entre les connaissances théoriques de la conception architecturale et les outils mis à la disposition des concepteurs nous a permis de dégager les conditions nécessaires pouvant permettre à un acteur participant à la conception architecturale de se servir d'un outil informatique dans le cadre de ses compétences.

Afin qu'un outil puisse contribuer à la conception, il doit répondre à ces deux conditions :

- Aux profils des différents concepteurs. Pour cela, il nous faut un outil offrant un dialogue interactif naturel et accessible selon les capacités informatiques des usagers. Pour y faire, nous avons opté pour un langage visuel qui permet à tout usager de l'outil informatique de dialoguer avec un ordinateur de la façon la plus simple.

- Aux intentions et aux besoins des différents concepteurs. Pour répondre à cette condition il faut disposer un outil qui puisse assister et répondre aux préoccupations de ces usagers. Pour y faire, nous avons opté pour l'approche procédurale qui permet de décrire une famille d'objet partageant des liens de parenté.

L'approche que nous avons proposée se base sur l'instrumentation du dialogue pouvant lier un acteur de la conception architecturale à son ordinateur en vue de générer un modèle d'objet architectural est validé puisque les deux caractères (validité interne et validité externe) qui permettent d'évaluer la qualité d'une stratégie de recherche retenue sont vérifiées.

Ainsi, la validité interne a été assurée par l'inexistante d'autres hypothèses rivales qui peuvent assurer la relation de cause à effet, autres que la solidité de nos hypothèses qui se basaient sur

la possibilité d'instrumenter les procédures de création des modèles d'objets architecturaux pour permettre à tout acteur de la conception architecturale de rendre visible ces objets sur le quel il veut communiquer.

La validité externe est assurée par la réussite des résultats liée à la possibilité de généraliser notre stratégie à d'autres modèles d'objets. En effet, la cohérence des résultats obtenus lors de notre expérimentation avec nos hypothèses de départ, nous laisse prétendre sur la possibilité de généraliser notre stratégie à d'autres contextes.

Cette approche ne constitue qu'un essai de dialogue pouvant lier un concepteur à son ordinateur. Des recherches plus approfondies sont nécessaires pour étudier la question en profondeur. Pour le développement d'un tel assistant, une étude approfondie est nécessaire pour rendre l'interface en conformité avec le mode de penser de tous les acteurs de conception est nécessaire.

IV.2) PISTES DE DEVELOPPEMENT :

La collaboration entre les différents acteurs de la conception architecturale continue de susciter le quotidien des recherches dans le domaine de la CAO. L'une de ces pistes de développement actuelle est la conception et la vente assistée par ordinateur (CVAO). Cette piste vise à concevoir en temps réel et en fonction de besoins particuliers d'un projet mais se limite seulement à certains éléments de constructions standards tels les cuisines, salles de bains, ...etc.

Cette piste prometteuse peut s'étendre à tous les éléments de constructions et permettra d'obtenir des ordres de coûts de revient de toute solution architecturale si on développerait des outils de co-conception assistée par ordinateur qui tiennent compte des avis de tous les métiers impliqués.

L'autre aspect des nouvelles voix de recherche dans le domaine de la CAO est celui qui permettrait la collaboration de tous les acteurs de conception sans leurs présences. Cette tendance ouvre la porte aux systèmes relationnels et mobiles où la présence physique n'aura plus sa place. Cette voix permettra aux acteurs concernés, la possibilité d'accès à tous les

dossiers relatifs à l'évolution d'un projet architectural et de participer en temps réel à certaines décisions de n'importe quel endroit où l'on se trouve.

IV. APPLICATIONS POSSIBLES :

À la lumière de ce qui précède, notre stratégie peut être utilisée à plusieurs fins d'usages notamment :

- Elle peut être utilisée dans le cas d'une encyclopédie d'architecture où habituellement les encyclopédies traditionnelles décrivent un modèle d'objet à travers le quel on explique une famille d'objets. Les usages de cette encyclopédie ne récupèrent que l'image mentale du modèle d'objet aux caractéristiques qu'ils souhaitent. Ainsi, pour décrire un escalier, l'encyclopédie traditionnelle se limite à une image d'un escalier à la quelle qu'on ajoute une explication textuelle.

  Avec notre méthode, l'usager aura à visualiser les objets avec les caractéristiques souhaitées. Pour visualiser un escalier, il suffira qu'il intervienne sur ses propriétés.

Bibliographie

- Achten, H.H. *Requirements for Collaborative Design in Architecture.* Timmermans, Harry (Ed.), Sixth Design and Decision Support Systems inArchitecturand Urban Planning - Part one: Architecture Proceedings Avegoor, the Netherlands), 2002

- Atlas d'architecture mondiale. , dictionnaire. Paris; édition : stock et Librairie générale française, 1978.

- Bâtie David. L, AIA , phd East california ; *the incorporation of construction history. Architectural history :the histcon . Interactive comptur program,* ACADIA 96.

- Boudon, Philippe. Pousin Frediric. , *Figures de la conception architecturale : Manuel de figuration graphique*. Paris. Édition Dunod, 1998.

- Boudon, Philippe, Introduction à l'architecturologie. Paris. Édition Dunod, 1992.

- Callon Michel, la conception architecturale, Editions parenthéses. Presse universitaire de France. P25-34. 1996

- Charbonneau Nathalie., *Méthode proposant l'utilisation de la programmation fonctionnelle pour la génération des solutions architecturales partielles sous forme matricielle.* Mémoire de maîtrise en CMFAO .Faculté d`aménagement. Université de Montréal, 2002.

-Contiandriopoulos André -Pierre, Champagne .Francois, Potvin Louise, Denis Jean-Louis Boyle Pierre, *Savoir préparer une recherche. La définir, la structurer, la financer,* Montréal ; les presses universitaires de Montréal, 1990

- Clayssen .D /.Lobstein .D /.Zeitoun .J, *Les nouvelles image,* Paris .Édition : Bordas 1990

- Couwenbergh. J – P, *La synthèse d'images*, Allur -Belgique édition : marabout informatique, 1998.

- Coyen. R. S. Mclaughlin , S.Newton, *Information technology and praxis : a survey of computers in design practice,* dans : Environment and planning b : Planning and design 1996. Coyne Richard, Rosenman, M.A. Radfort, A.D; Balachandran; Gero; J.S , *Knoledge-based design systems.* Addison-Wesley publishing, reading, MA. 1990.

-Dragomir.V, Gheorghiu .A, *Représentation géométriques des structures spatiales*, Luttre .Édition : Paul Mignot, 1969.

- De Paoli Giovani et Bogdan, Marius, *The front of the stage of vitruvius Roman théâtre*, dans Godfried Augenbroe et Charles Eatman (eds) CAAD futures99 Boston : kluwer academic pp 321-334

- De Paoli Giovanni, Pellissier Pierrlucio, *Dessin d'architecture par ordinateur,* Québec Édition : berger 1992

- De Paoli Giovanni, Tidafi Temy, *Modélisation architecturale et outils informatiques, entre cultures de la représentation et du savoir –faire,* Montréal édition :acfas 2000.

- De Paoli Giovanni : *une nouvelle approche à la conception par ordinateur en architecture basée sue la modélisation d'opérateurs sémantiques et la création de maquettes procuratrices.* Thèses de doctorat. Faculté d`aménagement. Université de Montréal 2000.

- Deshayes Catherine. *Processus interactionnel client /architecte en situation de conception de maison individuelles.* Plate forme Afia. Laval 2003. Article publié dans le journal (le monde, 3 mai, 2003, p.17).

- Dieu –Hanh Pho Anice, *description informatique de l'évolution historique d'un bâtiment – une modélisation de la transformation*. Montréal : Mémoire de maîtrise en CMFAO. Faculté d'aménagement. Université de Montréal 1997.

- Direction des cours par correspondance, *les escaliers, construction et rénovation* Québec : Les publications du Québec 1985.

- Dubuisson Bernard, *encyclopédie pratique de la construction du bâtiment*, Paris 7eme : Édition : Librairie Aristide Quillet 1995.

- Ellis, Deidre, *Le diagramme : description informatique d'opérateurs de transformation dans un processus de conception architecturale, mémoire* de maîtrise en CMFAO .Faculté d`aménagement. Université de Montréal 2001.

- Flemming, U, *More than the sum of parts : the grammar of Queen Anne houses*, Environnement and Planning B: Planning and design 1987, volume 14, pages 323-350.

- Guena François, *Le raisonnement par classification appliqué à la cao*. Université de Caen 1997.

- Halin Gilles, Bignon Jean-Claude, Nakapan Walaiporn, Humbert Pascal, Wagner Marc : *Batimage : la recherche d'informations techniques*. Article paru dans la revue : International Journal of Intelligent Systems, volume n° 19, pages 65/78, 2004. Wiley Publishers Since 1807.

-Halin Gilles, Bignon Jean-Claude, Hanser Damien, Malcurat Olivier. *COCAO : Modélisation d'un environnement logiciel coopératif pour les acteurs de l'architecture et B.T.P.* Conférence Internationale ICE 2002, Rome, juin 2002.

- Iordanova Ivanka, *les objets-types en conception architecturale et leur représentation par la modélisation du savoir-faire, mémoire* de maîtrise en CMFAO. Faculté d'aménagement. Université de Montréal2000.

- Joseph .A.wilkes, *encyclopedia of architecture*. New York . édition Willky 1960.

- Kalay , Y.E , Khemlani Lachmi –Timerman Anne – Beatrice Benne, *Semantically rich bulding représentation*. Dans J.Peter Jordan , Bettina Mehnert et Anton Harfmann(eds), representation in design ACADIA 97; Cincinnati Printing . pp 207- 227

- Khemlani Lachmi: Semantically Rich Bulding Représentation ._University of california at Berkeley -Acadia 1997.

- Fröst Peter. *A Real Time 3D Environment for Collaborative Design.* Digital Design - Research and Practice [Proceedings of the 10th International Conference on Computer Aided Architectural Design Futures / ISBN 1-4020-1210-1] Tainan (Taiwan) 13–15 October 2003, pp. 203-212.

- Larousse, *Le petit Larousse illustré*, Paris, édition Larousse 2001.

-Laurent Roger, De l'image naturelle à l'image artificielle. Paris. Editons l'harmattan. 1994.

- Le Moigne J.L, *Intelligence des mécanismes, mécanismes de l'intelligence*, Paris, Fayard, Fondation Diderot 1986.

- Le Moigne J.L, la modélisation des systèmes complexes ; Dunod, 1990.

- Le nouveau petit Robert, Dictionnaire alphabétique et analogique de la langue française. Paris. Le Robert, 2000

- Lipp .J-J, *Cours de méthodologie du design,* Faculté d'aménagement université de Montréal 1970.

- Loyer Michel, *La CAO le DAO* ; Paris. Presse universitaire de France. Collection : que sais je 1991.

- Mannes Willibald, *Technique de construction des escaliers*, Paris. Edition eyrolles 1980

- Marie-Eva Deville, Multi -dictionnaire de la langue française. Montréal. Édition Québec – Amérique 1997.

- Mitchell, William J, *Computer Aided Architectural Design*, Petrocelli/Charter , New York 1987.

- Mitchell, William J, *The art graphics programming : a structured introduction for architects and designers*, Van Nostrand Reinhold, New York, 1987.

- Mitchell, William J, *The logic of architecture: Design, computation and cognition* Cambridge ,Mass. M.I.T. Press, 1990.

- Mitchell L, William J. Mcculoogh, Malcolm, *Digital design media :A handbook for architects& Design professionals*. Van Nostrand Reinhold, New York 1991.

- Morin Edgar, *La méthode III, la connaissance de la connaissance.* Paris VI, Édition du Seuil 1986.

- Morin Edgar, *La méthode III, la connaissance de la connaissance.* Paris VI, Édition du Seuil 1986.

- Parisel Claude, Notes de recherches. Montréal ; université de Montréal 1990.

- Parisel Claude –Tidafi Temy, *le modèle architectural dans son contexte informatique, redéfinitions ou propositions*, bulletins d'information de l'école d'architecture de Paris – Villemin (EAPV) n 29 juin 1998 pp16- 19

- Prost. R, concevoir, *inventer, créer : les réflexions sur les pratiques*, Paris. Édition l'harmattan 1994.

- Prost Robert, Conception architecturale : une investigation méthodologique. Paris. Editions l'harmattan 1992.

- Quintraud .P/ Autran.J /Floren Zano.M / Fregier.M/Zoller.J (1985) : *La CAO en architecture,* Paris, édition : hermes 1985.

Ratier Corinne, sensibilisation à la démarche d'analyse de travail. Centre national de la recherche scientifique. Direction des systèmes d'information. Diffusion : Équipe projet DSI 2000.

- Robert Paul, *Dictionnaire de la langue française*, Paris 2eme. Édition : Société du nouveau Littré 1965.

- Sasada,Tsuyoshi, *Computer graphics as a communication medium in the design*. CAAD Futures 1995, p 3-5.

- Simon herbert. A : L*a science des systèmes, science de l'artificiel,* Paris .Éditeurs Épi .sa

- Shu Nan C, *visuel programming*, New York: Van Nostrand Reinhold Compagny 1988

- Stiny George, introduction to shape and shape grammars; dans : environnement and planning B, volume 08, p 343,351, 1980.

- Tidafi Temy, *moyens pour la communication en architecture proposition de la modélisation d'action pour la figuration architecturale,* thèse de doctorat ; faculté d`aménagement ; Université de Montréal 1996.

- Union nationale des syndicats français d'architectes (UNSFA), *l'architecte et l'informatique,* Paris ; édition : moniteur 1987.

Verrier. C, *vocabulaire de recherche*, Paris 8eme, DEA " Approches Plurielles en Sciences de l'Education " 1999.

- Walliser, Bernard, *Systèmes et modèles, introduction critique à l'analyse de systèmes*, Paris, Édition du soleil1977.

SITES INTERNET:

- Encyclopédie informatique : http://www.commentcamarche.net

ANNEXE

```
//*********************************************************************
/* Desc:  Programme en java permettant d'apporter des changements sur un programme en pov-ray et de lancer le modeleur pov-ray.

 Les changements concernés sont : la hauteur, le nombre de marche, la largeur de la marche et le type de marche d'un escalier

Date: 11/11/2004

Auth: Benmoumene Djebbar

*/

//*********************************************************************

import java.util.*;

import java.io.*;

public class Escalier

{

 static void lireCreer(String ficOrigine,String ficCree,int [] nbLigne,String var1,

String var2,String var3,String var4,int nouvelleValeur1,int nouvelleValeur2,

 int nouvelleValeur3,int nouvelleValeur4)throws IOException

    {

        boolean finFichier = false;

      boolean existeFichier = true ; // à ajuster après

      FileReader fr = null; // initialiser pour Java

                  FileWriter fileWriter = null;

                  PrintWriter pw = null;

                  int compteurLignes = 1;


       // essayer
```

```
 try {
      fr = new FileReader (ficOrigine) ;
     }
catch ( java.io.FileNotFoundException erreur) // intercepter l'erreur
    {
     System.out.println("Probleme d'ouvrir le fichier " + ficOrigine);
     existeFichier = false ; // ajuster
    }
try
  {
    fileWriter = new FileWriter(ficCree);
     pw = new PrintWriter(fileWriter) ;
  }
catch ( Exception erreur) //intercepter l'erreur
  {
     System.out.println("Probleme de creer le fichier " + ficCree);
     existeFichier = false ; // ajuster
    }

 if (existeFichier)
  {
   // un modèle de lecture d'un fichier:
   BufferedReader entree = new BufferedReader(fr);
     while ( !finFichier )
           {
          String ligneLue = entree.readLine();
```

```
 if (ligneLue == null)
 finFichier = true ;
else
    {
     if(compteurLignes == nbLigne[0])
       {
          int indice = ligneLue.indexOf(var1);
          String nouvelleLigne = ligneLue.substring(0,indice)+var1+" = "
          +nouvelleValeur1;
          pw.println(nouvelleLigne);
       }
  else if(compteurLignes == nbLigne[1])
     {
      int indice = ligneLue.indexOf(var2);
     String nouvelleLigne = ligneLue.substring(0,indice)+var2+" = "
     +nouvelleValeur2;
      pw.println(nouvelleLigne);
     }
 else if(compteurLignes == nbLigne[2])
    {
     int indice = ligneLue.indexOf(var3);
    String nouvelleLigne = ligneLue.substring(0,indice)+var3+" = "
    +nouvelleValeur3;
    pw.println(nouvelleLigne);
   }
  else if(compteurLignes == nbLigne[3])
```

```
                    {
                      int indice = ligneLue.indexOf(var4);
                      String nouvelleLigne = ligneLue.substring(0,indice)+var4+" = "
                      +nouvelleValeur4;
                       pw.println(nouvelleLigne);
                    }
                else
                    pw.println(ligneLue);
                     compteurLignes++;
              }
            }
            entree.close();
        }

                    fr.close();
                        pw.close();
}

public static void main (String[] args) throws IOException
 {
  BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));



  System.out.println("              *******  Escalier  ****");
  System.out.println("                 ----------------");
```

```
int [] lignes = {43,48,50,56};
System.out.println("choisir le type d'escalier : ");
 System.out.println(" ");
 System.out.println(" ecrire 1 : escalier balance ");
 System.out.println(" ecrire 2 : escalier colimacon");
 System.out.println(" ecrire 3 : escalier pyramidal");
 System.out.println(" ecrire 4 : escalier droit");
 System.out.println(" ");
 System.out.print("Entrer le type d'escalier : ");
int valeur1 = new Integer(stdin.readLine().trim()).intValue();

System.out.print("Entrez la hauteur de la  marche ( en cm) : ");
 System.out.print(" ");
 int valeur2 = new Integer(stdin.readLine().trim()).intValue();

 System.out.print("Entrer le nombre de marche : ");
 int valeur3 = new Integer(stdin.readLine().trim()).intValue();

 System.out.print("Entrer la largeur  de la marche ( en cm): ");
 int valeur4 = new Integer(stdin.readLine().trim()).intValue();

 lireCreer("Escalier.txt","ficCree30.txt",lignes,"typmarche","h","n","t1",
                        valeur1,valeur2,valeur3,valeur4);

 lireCreer("Escalier.txt","ficCree30.txt",lignes, "typmarche","h","n","t1",
                        valeur1,valeur2,valeur3,valeur4);
```

```
try {
    String[] command = new String[3];
   command[0] = "C:\\Program Files\\POV-Ray for Windows v3.6\\bin\\PVEngine.EXE";
  command[1] = "Mes documents\\Visual Studio Projects\\Project10\\ficCree34.pov";
  command[2] = "sortie.bmp";
  Runtime runtime = Runtime.getRuntime();
  Process proc = runtime.exec( command ); // lance l'execution
 }
   catch(FileNotFoundException e)
  {
   System.err.println("Could not open file ... ");
   throw e;
   }
 }
}
```

```
//*********************************************************************
// Desc: // description générale d'un escalier
// Date: 11/11/2004
// Auth: Benmoumene Djebbar
//*********************************************************************
background { color red 0.8 green 0.8 blue 0.7 }
//*********************************************************************
//Lumieres
 //*********************************************************************
light_source{
                0*x // light's position (translated below)
                color red 1.0  green 1.0  blue 1.0  // light's color
                 translate <-20, 40, -20>
               }
//*********************************************************************
//Camera
  //*********************************************************************
    camera {
               location <-5.5 ,6.5 , 6.5 >
               look_at  < 1, 1.5, 1>
              }
/*
            {
             location  <-3 ,4.5 , -5 >
            look_at  <- 0, 1.5, -2>
          }
```

```
        {
           location  <6 ,4 , -7.5 >
          look_at   < 0, -2, 0>
        }
      */
//**************************************************************************
//les textures
 //**************************************************************************
#include "colors.inc"
#include "textures.inc"
#include "shapes.inc"
#include "stones.inc"

//**************************************************************************
//Type de volée de marche
//**************************************************************************
  #declare typmarche = 3
   ;     // choix du type de marche
//**************************************************************************
//Les déclarations des points communs entre les types d'escaliers
//**************************************************************************
#declare h = 19
/100;// hauteur de la marche
#declare n = 6
; //nombre de marche première volée
declare n1 = 5 ;  //nombre de marche deuxième volée
```

```
declare n2 = 3 ;  //nombre de marche troisième volée
declare n3 = 5 ;  //nombre de marche quatrième volée
declare n7 = 8 ;  //nombre de marche de l'escalier balancé
#declare t1 = 2
/100 ;    // largeur de la marche
#declare g = 0.62 - (2*h)   ;
#declare escalier = 0 ;
# declare r = 0; //    rayon du poteau
#declare R = 0.6 + r;   // rayon    de l'axe de l'escalier
#declare m = t1 +r;
#declare  p1 = g/R ;
#declare  p =   (180*p1)/pi  ;
#declare  materiau =     T_Stone8
//*********************************************************************
// Les axes de repères
//*********************************************************************
 #declare axis =   union {
                          cylinder { <0, 0, 0>, <-50, 0, 0>, 0.025
                            pigment {color rgb <0,0,1>}
                          }
                          cylinder { <0, 0, 0>, < 0,50, 0>, 0.025
                            pigment {color rgb <0,1,0>}
                          }

                           cylinder { <0, 0, 0>, <0,0,-50>, 0.025
                              pigment {color rgb <1,0,0,>}
```

```
                    }
                }
 object  {axis}
//*************************************************************************
//Déplacement de l'escalier
//*************************************************************************
#declare p_dep=
union {

//*************************************************************************
// Procédure de création de la marche
//*************************************************************************
        #declare marche=  // la marche de départ
            difference{
                    box   {
                            <0, 0, 0>
                           < m,  h,  m>
                           texture { materiau}
                          }



//*************************************************************************
// Les plans de coupes
//*************************************************************************
      #declare planParallele = // plan de coupe de la marche
```

```
                                plane {x, 0.0}
    #declare planParallele1= // plan de coupe de la marche
                                plane { -x, 0.0 }
//*************************************************************************
// Procédure d'une volée  balancé type proportionnel
//*************************************************************************
          #if (typmarche = 1)
        }
  #declare Count = 0  ;
  #while (Count  < n)
  #macro Factorial(N)  //  fonction qui permet d'ajouter la meme part
  #local Result = 0;
  #local Ind = 1;
  #while(Ind <= N)
  #local Result = Result+Ind;
  #local Ind = Ind+1;
  #end
  Result
  #end

  #declare c1  = 0.2 ;// marches balancées
  #declare Value = Factorial(n);
  #declare  part  =  (g-c1)/(n);// part
  #declare Value1 = Factorial (Count-1);
  #declare Value2 = Factorial (Count-2);
```

```
    #declare   ang   = atan2 ((Count)*g-((Count)*g-(part*(Value1))),0.45);
    #declare   ang1   =  atan2 ((Count-1)*g-((Count-1)*g-
   (part*Value2)),0.45);

    difference{
                object {marche scale m*Count*x translate < 0 , h*Count ,0 >
               }
                object {plane{ x, -g+(Count)*part }rotate degrees(ang1)*y
                          translate< g+(g*(Count)-(Value1*part))-0.01,0>
                        }
               object {plane{-x, (Count-1)*part}rotate degrees(ang)*y
                        translate < g+(g*(Count)-(Value2*part)),0>
                       }
              }

   #declare Count =  Count+1 ;
   #end
   #end
//**********************************************************************
// Procédures d'une volée Colimaçon
//**********************************************************************
    #if (typmarche = 2)    // choix de l'escalier colimaçon
      object  {planParallele} // plan de coupe1
        object  {planParallele rotate -y*(180-p) }// plan de coupe2
                  cylinder {
                             -0.0001*y,  (n*h)*y,  r
```

```
                    }
                }
    #declare Count = 0;
    #while (Count < n )
    object   { marche rotate 0*y rotate y*p*Count translate y*h*Count }
    #declare Count = 1+ Count;
    #end
    #end
//*************************************************************************
// Procédures d'une volée pyramidale
//*************************************************************************
    #if (typmarche = 3)  // choix de l'escalier colimaçon
                }
    #declare Count = 0   ;
    #while (Count < n )
    difference{
                object{ marche translate< g*Count,h*Count,0> rotate 0*y  }
                object {plane {z, 0}translate < 0,0,g*Count > }
                 object {plane {-z, 0}       translate < 0,0,m-g*Count >  }
                 object {plane {-x 0}       translate <m,0,0 >  }
               }
     #declare Count =  Count+1 ;
    #end
    #end
   //*************************************************************************
```

```
//Procédures  d'une volée droite
//*********************************************************************

 #if (typmarche = 4)   // choix d'escalier droit

 #declare m1= 1.4

/100;//    m1 est la largeur du palier de repos

                    object  {planParallele }

                   object  {planParallele1  translate  <  g,  0,0>}

          }

   object {marche}

   #declare Count = 0    ;

   #while (Count  < n  )

   object   { marche translate < g*Count , h*Count ,0> rotate 0*y  }

   #declare Count = 1+  Count ;

   #end

//*********************************************************************

//Procédures de création de palais de repos
//*********************************************************************

  #declare rep =  0;  // choisir 0 pour le palier et 1 pour sans palier

  #if(  rep = 1)         //choisir 0 pour le palier et 1 pour sans palier

  #declare repos1=  // palais de repos

             box {

                  <(n-1)*g , (h*(n-1))  0>  // one corner position <X1 Y1 Z1>

                  <(n-1)*g+m ,h*(n-1)+h, (m1)> // other corner position <X2 Y2 Z2>

                  texture { T_Stone8}

                }

   object {repos1 rotate 0*y }

  #end
```

```
//*************************************************************************
// Escaliers à plusieurs volées
// (Ce que le langage visuel st censé réécrire)
//*************************************************************************
   #declare volee1 = // pour ajouter une volée
   #declare vol = 0;
   #if( vol = 1)
   #declare a1 = 0; // angle
   #declare marche1 =
   union {
            #declare Count = 0   ;
            #while (Count < n1 )
            object { marche translate < g*Count , h*Count ,0> rotate 0*y }
             #declare Count = 1+ Count ;
              #end
           }
  object { marche1 Rotate_Around_Trans( y*a1,< 0 ,0 ,0> ) }
  //*************************************************************************
// Direction de la volée
//Ajout d'une volée
//*************************************************************************
#declare repos2= // palais de repos
                box {
                     <(n1-1)*g , (h*(n1-1)) 0>
                    < (n1-1)*g+m , h*(n1-1)+h, (m) >
                    texture { materiau}
```

```
        }
        #if ( a1 = 0) // ajouter une volée droite directe
        object {volee1 translate < g*(n-1)+m , h*n ,0> }
        #end

        # if( a1 = 90) // ajouter une volée droite tout en tournant 90 degré
        object {volee1 translate < g*(n-1) , h*n ,0> }
        #end

        #if( a1 = -90) // ajouter une volée droite tout en tournant -90 degré
        object {volee1 translate < g*(n-1)+m , h*n ,m> }
        #end

        #if( a1 = 180)  // ajouter une volée droite tout en tournant 180 degré
        object {volee1 translate < g*(n-1) , h*n ,m1> }
        #end
  #end
  //*************************************************************************
// Paliers de repos
//*************************************************************************
#declare mrep = 0.8;
  #declare a1 = 0; // angle
   #declare rep = 0; // choisir 0 pour le palier et 1 pour sans palier
  #declare repos2= // palais de repos
          box {
              <(n1-1)*g , (h*(n1-1)) 0>
```

```
        < (n1-1)*g+m*rep ,  h*(n1-1)+h, (m) >
        texture { materiau}
    }

 #if(a1 = 0& rep = 0) //choisir 0 pour le palier et 1 pour sans palier
 object {repos2 Rotate_Around_Trans( y*a1,< 0 ,0 ,0> )  translate  < g*(n-1)+m ,

          h*n ,0> }
 #end

 #if( a1 =  180& rep = 1)//choisir 0 pour le palier et 1 pour sans palier
object {repos2 Rotate_Around_Trans( y*a1,< 0 ,0 ,0> ) translate  < g*(n-1) , h*n
         ,m1> }
 #end

 #if(  a1 =  -90& rep = 1)//choisir 0 pour le palier et 1 pour sans palier
 object {repos2 Rotate_Around_Trans( y*a1,< 0 ,0 ,0> )  translate  < g*(n-
         1) +m*rep , h*n ,m*rep> }
 #end

 #if(  a1 =  90& rep = 1) //choisir 0 pour le palier et 1 pour sans palier
 object {repos2 Rotate_Around_Trans( y*a1,< 0 ,0 ,0> )  translate  < g*(n-1) , h*n
         ,0>  }
#end
```

```
  #end
}
object {p_dep /*translate < -7, (3+0.002)*h, -2.05 >  */rotate 180*y}
```

```
//*****************************************************************
// Desc: Projet du glass house fait par Bruno TAut
// Date: 11/11/2004
// Auth: Benmoumene Djebbar
//*****************************************************************
light_source {
                0*z     // light's position (translated below)
                color rgb <0.51,0.51,0.51>   // light's color
                translate <-3, 4.2,0>
            }
light_source {
                0*z  // light's position (translated below)
                color rgb <0.51,0.51,0.51>   // light's color
                translate <-6, 3.4,1.5>
             }
light_source {
                0*z  // light's position (translated below)
                color rgb <0.51,0.51,0.51>   // light's color
                translate <-5, 3.8, -1.5>
            }


light_source {
                0*z  // light's position (translated below)
                color rgb <0.21,0.21,0.251>   // light's color
                translate <-4, 3.4, 0>
```

```
                }
light_source {
                 0*z  // light's position (translated below)
                 color rgb <0.21,0.21,0.21>   // light's color
                 translate <-7, 4, 0>
               }
light_source {
                  0*z   // light's position (translated below)
                  color rgb <0.41,0.41,0.41>    // light's color
                 translate < -4.6,4.5,3.9>rotate -31*y
               }
light_source {
                 0*z  // light's position (translated below)
                 color rgb <0.41,0.41,0.41>    // light's color
                 translate < -3.6,4.5,-3.9>
               }
 //*************************************************************************
                          // Inclure les textures
//*************************************************************************
 #include "colors.inc"
#include "skies.inc"
  sky_sphere { S_Cloud5 }
  #include "functions.inc"
#include "logo.inc"
#include "colors.inc"
#include "textures.inc"
```

```
#include "shapes.inc"
#include "stones.inc"
 #include "metals.inc"
 #include "glass.inc"
 #include "metals.inc"
 #include "glass.inc"
//*************************************************************************
                              //  Inclure les escaliers sans les decrire
 //*************************************************************************
 #include "esc-pyramide.inc"
#include "escalier-principal.inc"
  #include "esccolimacon.inc"
#include "balance-d1.inc"
#include "bal-gauche.inc"
#include "colimac.inc"
//#include "escArDroit1.inc"
 #include "escArGauche1.inc"
#include "colimacon-int-droite.inc"     //devant droite
#include "colimacon-int-gauche.inc"     //devant gauche
//*************************************************************************
                                    // cameras
//*************************************************************************
  camera
     {
       location <-20 ,6  ,-8.5>
       look_at  <-2 , 6 , 0.0>
```

```
    }
/*
    {
            location  <-20 ,6  ,-8.5>
            look_at   <-2 , 6 , 0.0>
    } */
//*************************************************************************
 #declare a= 24   ;  // angle entre les poteaux
#declare d = 180*0.3/(pi*5.5)   ;
 //*************************************************************************
                              //   Les axes de repères
//*************************************************************************
#declare axis =
      union{
              cylinder { <0, 0, 0>, <-50, 0, 0>, 0.025
                        pigment {color rgb <0,0,1>  }
                      }
             cylinder { <0, 0, 0>, < 0,50, 0>, 0.025
                        pigment {color rgb <0,1,0>}
                      }
             cylinder { <0, 0, 0>, <0,0,-50>, 0.025
                        pigment {color rgb <1,0,0,>}
                      }
            }
object  {axis}
//*************************************************************************
```

```
                                        // Socle ou sol
//*************************************************************************
declare sol=
        box{
            <-100, -0, -100>  // one corner position <X1 Y1 Z1>
            < 100,  -10,  100>  // other corner position <X2 Y2 Z2>
            pigment  { rgbf  < 0.28,0.278,0.29   >}
          }
object  {sol}
//*************************************************************************
                                    // Le soubassement
//*************************************************************************
declare base =      // cylindre de base (surelevation
        union{
              difference {
                          cylinder { 0*y,  19*h*y,  7.5 } // cylindre de base
                          box { // Parallepipéde utilisé pour l'escalier
                                <0, 0, 0>        <2.25, 8, 3 >
                                translate < -7.5, -0.2, -1.5>
                              }
                          torus{  // utilisé pour soustraire la fin du cylindre
                                  7.5,  0.4
                                  translate 19*h*y
                               }
                          box { // Parallélépipède utilisé pour l'escalier
                                <1.25, -2,2>   < 15, 24*h, -2 >
```

```
        }
}
  prism {
         linear_sweep
        linear_spline
       0, // sweep the following shape from here ...
       -0.2, // ... up through here
        4, //  number of points making up the shape
        <1.25,0.8>, <1.25,2>, <1.45,2>, <1.25,0.8>
        translate 19*h*y
        texture {pigment
        {rgbf <0.95, 0.95, 1.00, 0.1>*1.2}
        normal {bumps 0.1}
         }
}
   prism {
         linear_sweep
         linear_spline
         0, // sweep the following shape from here
        -0.2, // ... up through here
          4, // number of points making up the shape
       <1.25,-0.8>, <1.25,-2>, <1.45,-2>,
       <1.25,-0.8>
       translate 19*h*y
       texture {pigment
                {rgbf <0.95, 0.95, 1.00, 0.1>*1.2}
```

```
                    normal {bumps 0.1}
                }
            }
        }

 object{base   translate 0.0001*h*y
        texture {
            pigment {rgbf <0.45, 0.45, 0.45, 0.1>*1.2}
            normal {bumps 0.1}
        }
    }
//**********************************************************************
                    //Murs de l'escalier  principal
//**********************************************************************
#declare cagedroite=    //Murs de l'escalier  principal
        box{  // Parallélépipède utilisé pour inclure de l'escalier
            <0, 0,0>  // one corner position <X1 Y1 Z1>
            < -13*g , 20*h, 0.2 >
            }
object{ cagedroite pigment  {rgbf <0.45, 0.45, 0.45, 0.1>*1.2}
                    translate < -7.4+10*g , 0, -1.5 > }
object{ cagedroite pigment  {rgbf <0.45, 0.45, 0.45, 0.1>*1.2}
                    translate <  -7.4+10*g, 0, 1.45 > }
//**********************************************************************
                    //Arrondi   de la base
//**********************************************************************
```

```
declare arrondi=
        difference{

                    torus { // volume ajouté pour donner la courbure au cylindre
                            7.10,
                            0.4
                            translate (19*h-0.4)*y
                            }
                    box { // Parallélépipède utilisé pour inclure de l'escalier
                            <0, 0, 0>
                            <3, 4, 3 >
                            translate < -7.5, -0.2, -1.5>
                          }
                    box { // Parallélépipède utilisé pour inclure de l'escalier
                            <2.5, -2,2.2>
                            < 15, 20*h, -2.2 >
                            }
                  }
object {arrondi pigment {rgbf <0.45, 0.45, 0.45, 0.1>*1.2} }
//*****************************************************************************
                //poteaux
//*****************************************************************************
declare poteau=
            cylinder{
                        19*h*y, (19*h+3.5) *y, 0.125
                        pigment {rgb< 0.9 ,0.9 , 0.9>}
```

```
                    }
# declare Count = 0 ;
 #while (Count  <6)
 object   {  poteau translate <-5.25, 0 ,-1.2>  texture {  Soft_Silver  }
                    rotate -y* a*Count }
 #declare Count = 1+  Count   ;
 #end
# declare Count = 0  ;
 #while (Count  <6)
 object   {  poteau translate <-5.25, 0 ,1.4>  texture {  Soft_Silver  }  rotate y*
a*Count}
 #declare Count = 1+  Count  ;
 #end
//*************************************************************************
                                    //Poutre
//*************************************************************************
declare poutre =
    difference{
              cylinder {
                        (19*h+3.5)*y, (19*h+3.9) *y,  5.55
                       }
             cylinder{
                        (19*h+3)*y, (19*h+4) *y,  5.30
                      }
            }
object {poutre  texture {Soft_Silver  }  }
```

```
//*************************************************************************
                    //Mur de séparation entre les escaliers int.
//*************************************************************************
declare murSepEsc=
        difference  {
                     cylinder{
                               (19*h)*y, (19*h+3.9) *y,  4.2
                              }
                     cylinder{
                               (18*h)*y, (19*h+5) *y,  4.1
                              }
                     object  { plane  {  -x, 0.0 } rotate -10*y    }
                     object  { plane  {  -z, 0 } rotate -16*y    }
                    }
object { murSepEsc rotate -12 *y  texture { T_Stone8 }}
 object { murSepEsc   texture { T_Stone8 } rotate -222 *y}
//*************************************************************************
            // Demi cercle du mur de séparation entre les escaliers int.
//*************************************************************************
declare demicercle=
        difference{
                    cylinder {
                               (19*h)*y, (19*h+3.9) *y,  0.7
                              }
                   cylinder  {
                              (19*h)*y, (19*h+4.9) *y,  0.6
```

```
                    }
                object  { plane  {  -z, 0.0 } rotate 40*y}}
object { difference{ object  {demicercle translate <-2.8, 0 ,2.1>
                        texture { T_Stone8 }}
                  cylinder {
                        19*h*y, (19*h+4.9) *y,  3.1
                        }
              } rotate -0*y
    }
object { difference{ object  {demicercle translate <-2.8, 0 ,2.1>
                        texture { T_Stone8 }}
                  cylinder {
                        19*h*y, (19*h+4.9) *y,  3.1
                        }
              } rotate 216*y
    }

//*********************************************************************
 // Demi cercle du mur de séparation entre les escaliers int.
//*********************************************************************
 declare demicercle=
        difference {
                  cylinder{
                        (19*h)*y, (19*h+3.9) *y,  0.7
                        }
                  cylinder {
```

```
                                (19*h)*y, (19*h+4.9) *y,  0.6
                              }
                        object  { plane  {  z, 0.0 } rotate 40*y }
                      }
object { difference{ object  {demicercle translate <-2.8, 0 ,2.1>
                                texture { T_Stone8 }}
                      cylinder {
                                19*h*y, (19*h+4.9) *y,  3.1
                              }
                    } rotate 82*y
                  }
object { difference{ object  {demicercle translate <-2.8, 0 ,2.1>
                              texture { T_Stone8 }}
                            cylinder{
                                    19*h*y, (19*h+4.9) *y,  3.1
                                  }
                  } rotate 290*y
      }
//*********************************************************************
                        //les  barodages a droite et gauche
//*********************************************************************
declare barodage=
          cylinder{
                    19*h*y, (19*h+2.5) *y,  0.03
                  }
# declare Count = 1 ;
```

```
#while (Count <16)
object  { barodage translate <-5.15, 0 ,-1.35> texture { T_Silver_5A }
                    rotate -y* 3/2*Count}
object  { barodage translate <-5.15, 0 ,1.55> texture { T_Silver_5A }
                    rotate y* 3/2*Count}
#declare Count = 1+ Count;
#end
//*************************************************************************
declare centrebarodage=
            cylinder {
                19*h*y, (19*h+2.5) *y,  0.06
                }
object   { centrebarodage translate <-5.15, 0 ,-1.35> texture { T_Silver_5A }
rotate -y*a/2}
object   { centrebarodage translate <-5.15, 0 ,1.25> texture { T_Silver_5A }
rotate y* a/2}
//*************************************************************************
                        //poutre du barodage
//*************************************************************************
declare poutrebarodage= // poutre séparent le dôme et le corps
                    box { // diminuer le tore au niveau des escaliers
                        <-5.25, 19*h+2.5,-1.05>
                        < -5.28, 2.5+19*h-0.2,1.10>
                        }
object {poutrebarodage  texture { T_Silver_5A } rotate -y*a}
object {poutrebarodage  texture { T_Silver_5A } rotate y*a}
```

```
//*************************************************************************
                                //Mur intérieur
//*************************************************************************
declare interieur=
        difference{

                    cylinder {
                              19*h*y, (19*h+3.9) *y,  3.1
                             }
                    cylinder {
                              18*h*y, (19*h+4.5) *y,  3
                             }
                   box { // Parallélépipède utilisé pour inclure de l`escalier
                         <0, 0, 0>  // one corner position <X1 Y1 Z1>
                         < 10, 55.5, 10 >  // other corner position <X2 Y2 Z2>
                         rotate 45*y
                       }
                   box { // Parallepipéde utilisé pour inclure de l`escalier
                        <0.9, 19*h -8>
                        < 1.9, 19*h+2.5, 8 >
                       }
                }

object{interieur  texture {
                           pigment {rgbf <0.95, 0.95, 1.00, 0.1>*1.2}
                          normal {bumps 0.1}
```

```
                    }
    }
//*********************************************************************
                        // Vide sur dalle
//*********************************************************************
declare videsurdalle=
            difference{
                    cylinder {
                            (19*h+2.95)*y, (19*h+5.05) *y,  5.55
                            }
                    cylinder {
                            (19*h+2.5)*y, (19*h+6.05) *y,  3
                            }
                    object  { plane  {  -x, 0.0 } rotate -6*y    }
                    object  { plane  {  -z, 0.0 } rotate -50*y    }
                }
//*********************************************************************
                        // dalle
//*********************************************************************
declare dalle=
    difference {
                cylinder {
                        (19*h+3.75)*y, (19*h+3.9) *y,  5.55
                        }
                cylinder  {
                        (19*h+4.95)*y, (19*h+5.05) *y,  1.25
```

```
                    }
                object{videsurdalle rotate y*20  }
                object{videsurdalle rotate y*-218  }
            }
 object {dalle   pigment{ White}}

//*********************************************************************
                        // Mur extérieur en brique
//*********************************************************************
declare murext =
        difference {
                cylinder{
                        (19*h)*y, (19*h+3.5) *y,  5.5
                    }
                cylinder {
                        (18*h)*y, (19*h+4) *y,  5.45
                        }
                object  { plane  {  -z,0 } rotate (-3*a+a/2)*y    }
                object  { plane  {  z, 0} rotate (-3*a-a/2)*y    }
            }
# declare Count = 0 ;
#while (Count  <5)
object   { difference {object{   murext texture { T_Vicksbottle_Glass   }
                        rotate y*-a*Count }
                box {  // Parallélépipède pour inclure de l'escalier arrière
                        <0.5, -2,2.2>
```

```
                    < 15, 20, -2.2 >
                }
            }
        }
    object  { difference {object{   murext texture { T_Vicksbottle_Glass   }
                    rotate y*-a*Count }
                box { // Parallélépipède pour inclure de l'escalier arrière
                    <0.5, -2,2.2>
                    < 15, 20, -2.2 >
                }
            } rotate -6*a*y
        }
#declare Count = 1+  Count ;
#end
declare s_esc=
    difference {
            cylinder {
                (19*h)*y, (21*h) *y,  5.5
            }
            cylinder {
                (18*h)*y, (25*h) *y,  5.45
            }
            object  { plane  {  -z,0 } rotate (a/2)*y    }
            object  { plane  {  z, 0} rotate (-a/2)*y    }
        }
# declare Count = 0 ;
```

```
#while (Count  <10)
object{
        difference{
                object {
                        object { s_esc  rotate y*((-d))*Count  translate
                                (h)*y *Count texture
                                { T_Vicksbottle_Glass   }
                                } rotate 0*a*y
                        }
                object  {
                        plane  {  z,0 } rotate (-a/2)*y
                        }
                }rotate -2*a*y
        }
object {
        difference{
                object{
                        object { s_esc  rotate y*((d))*Count  translate
                                (h)*y *Count texture { T_Vicksbottle_Glass   }
                                } rotate 0*a*y
                        }
                object  {
                        plane  {  -z,0 } rotate (a/2)*y
                        }
                }rotate 2*a*y
        }
```

```
#declare Count = 1+  Count ;
#end


    //*************************************************************************
                                        //Cascade arriere
//*************************************************************************
# declare   cascade  =
     difference{
                union{
                      difference{
                                box { // Parallélépipède utilisé pour  l'escalier
                                    <0.9, -0,0.6>
                                    < 6.6, 19*h, -0.6 >
                                   }
                                box {  // Parallélépipède utilisé pour  l'escalier
                                    <5.7, 5*h, 8>
                                    < 7.5, 24*h,-8 >
                                   }
                                box { // Parallélépipède utilisé pour l'escalier
                                    <4.8, 8*h, 8>
                                    < 6, 24*h, -8 >
                                   }
                               box { // Parallélépipède utilisé pour l'escalier
                                    <3.9, 11*h, 8>
                                    < 6, 24*h, -8 >
```

```
                    }
                box { // Parallélépipède utilisé pour l'escalier
                    <3, 14*h, 8>
                    < 4.9, 24*h, -8 >
                    }
                box { // Parallélépipède utilisé pour l'escalier
                    <2.1, 17*h, 8>
                    < 3.9, 25*h, -8 >
                    }
            }
        box{  // Parallélépipède utilisé pour l'escalier
            < 0.9, 0, 0.77>
            < 1.68, 19*h, -0.77 >
        }
    }
  cylinder {
        (14*h)*y, (25*h) *y,  1.2
        translate <1.9, 0,1.80>
    }
  cylinder {
        (14*h)*y, (25*h) *y,  1.2
        translate <1.9, 0,-1.80>
    }
}
object {  cascade  pigment {
                    bozo
```

```
                color_map {
                        [0.00, rgb <0.35, 0.58, 0.88>*1.0]
                        [0.25, rgb <0.35, 0.58, 0.88>*1.1]
                        [0.50, rgb <0.35, 0.58, 0.88>*0.9]
                        [0.75, rgb <0.35, 0.58, 0.88>*1.0]
                        [1.00, rgb <0.35, 0.58, 0.88>*0.8]
                    }
            }
    }

//*****************************************************************************
                        //mur de cascade
//*****************************************************************************
declare murdeCascade =
    union{
        difference{
            cylinder {
                (19*h)*y, (19*h+0.9) *y,  1.1
                }
            cylinder {
                (15*h)*y, (29*h+0.9) *y,  1
                }
            object  { plane  {  -x, 0.0 }    }
        }
        box { // Parallélépipède utilisé pour inclure de l'escalier
            <0, 19*h, 1.1> // one corner position <X1 Y1 Z1>
```

```
                < 1.05, 19*h+0.9, 1>  // other corner position <X2 Y2 Z2>
                }
            box { // Parallélépipède utilisé pour inclure de l'escalier
                <0, 19*h,-1.1 >
                < 1.05, 19*h+0.9, -1 >
                }
        }
object {  murdeCascade   pigment {
                                    gradient z
                                    color_map {
                                                [0.00, rgb <0.01, 0.59, 0.81>]
                                                [0.70, rgb <0.01, 0.59, 0.81>]
                                               }
                                    frequency 4
                                  }
    }
//*********************************************************************
                                //mur2 de cascade
//*********************************************************************
declare murcascade1=
      union
            {
            box { // Parallélépipède utilisé pour inclure de l'escalier
                <5.7, 5*h, 0.65>
                < 6.6, 5*h+0.9, 0.55 >
               }
```

```
            box { // Parallélépipède utilisé pour inclure de l'escalier
                  <4.8, 8*h, 0.65>
                  < 5.7, 8*h+0.9, 0.55 >
                }
            box { // Parallélépipède utilisé pour inclure de l'escalier
                  <3.9, 11*h, 0.65>
                  < 4.8, 11*h+0.9, 0.55 >
                  }
            box { // Parallélépipède utilisé pour inclure de l'escalier
                  <3, 14*h, 0.65>
                  < 3.9, 14*h+0.9, 0.55 >
                }
            box { // Parallélépipède utilisé pour inclure de l'escalier
                  <2.1, 17*h, 0.65>
                  < 3, 17*h+0.9, 0.55 >
                }
            box { // Parallélépipède utilisé pour inclure de l'escalier
                   <1.9, 19*h, 0.65>
                   < 2.1, 19*h+0.9, 0.55   >
                  }
        }
object {murcascade1 translate <0, 0,0.05>
        pigment {
                  gradient z
                 color_map {
                             [0.00, rgb <0.01, 0.59, 0.81>]
```

```
                    [0.70, rgb <0.01, 0.59, 0.81>]
                }
            frequency 4
        }
    }
object {murcascade1 translate <0, 0,-1.25>
        pigment {
            gradient z
            color_map {
                [0.00, rgb <0.01, 0.59, 0.81>]
                [0.70, rgb <0.01, 0.59, 0.81>]
            }
            frequency 4
        }
    }


//*********************************************************************
                // petit cercle de la cascade
//*********************************************************************
 declare arond=
        difference {
            cylinder {
                (19*h)*y, (19*h+0.9) *y,  1.2
            }
            cylinder {
```

```
                    (14*h)*y, (22*h+0.5) *y,  1.1
                  }
              object  { plane  {  -x, 0 } rotate -0*y    }
               object  { plane  {  z, 0 } rotate 40*y    }
            }

object  { arond   translate <1.9, 0,-1.8>
        pigment {
                  gradient z
                   color_map {
                              [0.00, rgb <0.01, 0.59, 0.81>]
                               [0.70, rgb <0.01, 0.59, 0.81>]
                            }
                    frequency 4
                 }
        }
declare arond2=
      difference{
                  cylinder {
                             (19*h)*y, (19*h+0.9) *y,  1.2
                           }
                cylinder {
                           (14*h)*y, (22*h+0.5) *y,  1.1
                          }
                 object  { plane  {  -z, 0 } rotate -40*y    }
                 object  { plane  {  -x, 0 } rotate 0*y    }
```

```
                }

 object  { arond2   translate <1.9, 0,1.80>
            pigment {
                        gradient z
                        color_map {
                                    [0.00, rgb <0.01, 0.59, 0.81>]
                                    [0.70, rgb <0.01, 0.59, 0.81>]
                                  }
                       frequency 4
                     }
         }
 //**********************************************************************
                                    /// le dome
 //**********************************************************************
declare dome =
            union{
                  cylinder  {   // élément du premier niveau
                              <-5.25, (19*h+3.9),-1.1>
                              <-5.25, (19*h+5.5) ,-0> ,0.1
                             }
                  cylinder { //élément du premier niveau
                              <-5.25, (19*h+3.9) ,1.1>
                              <-5.25, (19*h+5.5) ,0> ,0.1
                            }
 //**********************************************************************
```

```
cylinder {    // élément du deuxième niveau
          <-5.25, (19*h+5.5),-0>
          <-4.8, (19*h+6.8),-1.03>,0.1
         }
cylinder {    // élément du deuxième niveau
          <-5.25, (19*h+5.5) ,-0>
          <-4.8, (19*h+6.8) ,1.03>,0.1
         }
//*****************************************************************
cylinder {    // élément du troisième niveau
          <-4.8, (19*h+6.8) ,-1.0>
          <-4.3, (19*h+8) ,0>,0.1
         }
cylinder {    // élément du troisième niveau
          <-4.8, (19*h+6.8) ,1.0>
          <-4.3, (19*h+8) ,0>,0.1
         }
//*****************************************************************
cylinder {   //élément du quatrième niveau  du dôme
          <-4.3, (19*h+8) ,0>
          <-3.8, (19*h+8.95) ,-0.83>        ,0.1
         }
cylinder {   //élément du quatrième niveau  du dôme
          <-4.3, (19*h+8) ,0>
          <-3.8, (19*h+8.95) ,0.83>         ,0.1
         }
```

```
//*************************************************************************
        cylinder  { // élément du cinquième niveau  du dôme
                    <-3.8, (19*h+8.95),-0.78>
                    <-3, (19*h+9.8),-0.>       ,0.1
                  }
        cylinder {   // élément du cinquième niveau   du dôme
                   <-3.8, (19*h+8.95) ,0.78>
                   <-3, (19*h+9.8) ,-0>       ,0.1
                   }

//*************************************************************************

        cylinder {   //élément du sixième niveau  du dôme
                    <-3, (19*h+9.8),-0>
                    <-2.2, (19*h+10.5),-1.>        ,0.1
                  }
        cylinder  {  //élément du sixième niveau  du dôme
                    <-3, (19*h+9.8),-0>
                    <-2.2, (19*h+10.5) ,1>         ,0.1
                 }
//*************************************************************************
        cylinder  {  //élément du dernier niveau du dôme
                    <-2.2, (19*h+10.5) ,-1>
                    <0, (19*h+11.8) ,0>          ,0.1
                   }
```

```
        }
//*************************************************************************
# declare Count = 1 ;
 #while (Count <16)
 object  {  dome texture { Soft_Silver}  rotate -y* a*Count rotate 180*y }
 #declare Count = 1+  Count ;
 #end

//*************************************************************************
                              //couverture du dome
//*************************************************************************
declare couvredome=
                union{
                      polygon {    //premier triangle
                                    3,
                                    <-5.25, (19*h+3.95) ,-1.2>
                                    <-5.25, (19*h+5.5) ,-0>
                                    <-5.25, (19*h+3.95) ,1.2>
                                  }
                      polygon {     //deuxieme triangle
                                    3,
                                    <-5.25, (19*h+5.5) ,-0>
                                    <-4.8, (19*h+6.78) ,-1.10>
                                    <-4.8, (19*h+6.78) ,1.10>
                                  }
                      polygon {  //troisième triangle
```

```
            3,
            <-4.8011, (19*h+6.78) ,-1.101>
            <-4.3, (19*h+8) ,0>
            <-4.8, (19*h+6.78) ,1.101>
          }
polygon {   //quatrième triangle
            3,
            <-4.3, (19*h+8) ,0>
            <-3.8, (19*h+9),-0.9>
            <-3.8, (19*h+9) ,0.9>
          }
polygon {   //cinquième triangle
            3,
            <-3.8, (19*h+9) ,-0.9>
            <-3, (19*h+9.8) ,-0>
            <-3.8, (19*h+9) ,0.9>
          }
polygon {   //sixieme triangle
            3,
            <-3, (19*h+9.8) ,-0>
            <-2.52, (19*h+10.2) ,-0.42>
            <-2.52, (19*h+10.2) ,0.42>
          }
polygon {   //septieme triangle
            3,
            <-2.52, (19*h+10.2) ,-0.42>
```

```
                    <-2.52, (19*h+10.2) ,0.42>
                    <-2.5, (19*h+10.5) ,-0.1>
                }
            }
# declare Count = 1 ;
#while (Count <16)
object { couvredome texture {T_Vicksbottle_Glass }
        rotate -y* a*Count   rotate y* 180*y
    }
#declare Count = 1+ Count ;
#end

//*********************************************************************
    declare couvredome1=
        union{
            polygon {    //premier renversé triangle
                3,
                <-5.25, (19*h+3.9) ,-1.2>
                <-5.25, (19*h+5.5) ,-0>
                <-4.8, (19*h+5.5) ,-2.1>
            }
            polygon {    //deuxième renversé triangle
                3,
                <-5.25, (19*h+5.35) ,-0>
                <-4.8, (19*h+5.35) ,-2.1>
                <-4.8011, (19*h+6.8) ,-1.101>
```

```
        }
polygon {    //troisième renversé triangle
         3,
         <-4.8011, (19*h+6.8),-1.101>
         <-4.3, (19*h+8) ,0>
         <-4.0, (19*h+8) ,-1.6>
        }
polygon {    //quatrième renversé triangle
         3,
         <-4.30, (19*h+8) ,0>
         <-3.8, (19*h+9),-0.9>
         <-4.0, (19*h+8) ,-1.6>
        }
polygon {    //cinquième renversé triangle
         3,
         <-3, (19*h+9.8),-0>
         <-3.8, (19*h+9),-0.9>
         <-2.8, (19*h+9.8) ,-1.1>
        }
polygon {    //sixième renversé triangle
         3,
         <-3, (19*h+9.8) ,-0>
         <-2.6, (19*h+10) ,-0.55>
         <-2.8, (19*h+9.8) ,-1.1>
        }
polygon {    //cinquième triangle
```

```
                    3,
                    <-2.2, (19*h+10.56) ,-1>
                    <-2.6, (19*h+10) ,-0.55>
                    <-2.4, (19*h+10.56) ,-0.12>
                  }
                polygon {    //sixieme triangle
                    3,
                   <-2.2, (19*h+10.56) ,-1>
                  <0, (19*h+11.8) ,0>
                 <-2.4, (19*h+10.56) ,-0.12>
                }
            }
# declare Count = 1 ;
 #while (Count  <16)
  object  { couvredome1  texture  {T_Vicksbottle_Glass }
            rotate -y* a*Count rotate 180*y
          }
  #declare Count = 1+  Count ;
 #end
//*********************************************************************
                          // Entrée arriere
//*********************************************************************
  #declare  boite=
          union{
                difference{
                      box {
```

```
                <2.2,0*h, -2.1  >
                < 7, 19*h+3.5, 2.1 >
            }
        box  {
                <2.2,0*h, -2 >
                < 6.9, 18*h+3.4, 2 >
            }
    }
difference{
        box {
                < 7, 0, 2.1 >
                < 10, 19*h+3, -2.1 >
            }
        box {
                < 6.5, 0, 2 >
                < 9.9, 19*h+3, -2 >
            }
    }
difference{
        box  {
                < 10, 0, 1.3 >
                < 14, 19*h+2.5, -1.3 >
            }
        box  {
                < 9.5, 0, 1.2 >
                < 13.9, 19*h+2.4, -1.2 >
```

```
                    }
                }
            }
        object {boite  texture {   pigment {rgbf <0.95, 0.95, 1.00, 0.1>*1.2}
                                   normal {bumps 0.1}
                               }
            }
```